PROTEGIENDO TU RED: CÓMO DETECTAR Y PREVENIR INTENTOS DE ACCESO NO AUTORIZADOS A TRAVÉS DE LDAP

PROTECTING YOUR NETWORK: HOW TO DETECT AND PREVENT UNAUTHORIZED ACCESS ATTEMPTS VIA LDAP

TRABAJO FIN DE GRADO
CURSO 2023-2024

AUTOR
SERGIO ATIENZA FUERTES

DIRECTORES
LUIS MARÍA COSTERO VALERO
YOUSSEF EL FAQIR EL RHAZOUI

DOBLE GRADO EN INGENIERÍA INFORMÁTICA-ADE
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Resumen

Protegiendo tu red: cómo detectar y prevenir intentos de accesos no autorizados a través de LDAP

El uso del protocolo LDAP puede facilitar en gran medida la gestión de cuentas de usuarios en una red, al permitir la centralización de las mismas en un servidor. De este modo un administrador podrá controlar el acceso de los usuarios a los diferentes equipos de forma sencilla, evitando la complejidad de gestión que supone la creación de cuentas locales en cada equipo. Pese a estas ventajas, LDAP tiene importantes problemas que deben ser tenidos en cuenta para trabajar con él, en especial, problemas de seguridad. LDAP no ofrece ningún método claro para detectar posibles ataques y comportamientos sospechosos en la red, como por ejemplo el acceso de un usuario a un equipo ajeno al que le corresponde. Para enfrentar estos problemas, se ha desarrollado una solución que permita a un administrador controlar los accesos producidos en los equipos, registrando en cada acceso la información más relevante que pueda requerir. De este modo se podrán detectar comportamientos potencialmente maliciosos que afecten a la red y tomar medidas preventivas en consecuencia.

Palabras clave

LDAP, Gestión de cuentas de usuario, Seguridad en redes, Control de accesos, Prevención de ataques, Auditoría.

Abstract

Protecting your network: how to detect and prevent unauthorized access attempts via LDAP

The use of the LDAP protocol can greatly facilitate the management of user accounts on a network, by allowing them to be centralized on a server. This allows an administrator to control user access to different computers in a simple way, avoiding the management complexity of creating local accounts on each computer. Despite these advantages, LDAP has significant problems that must be taken into consideration when working with it, especially security issues. LDAP does not offer any clear method for detecting possible attacks and suspicious behavior on the network, such as a user accessing a computer other than the one to which he/she belongs. To address these problems, a solution has been developed that allows an administrator to control the accesses produced on the computers, recording the most relevant information required for each access. In this way, potentially malicious behavior affecting the network can be detected and preventive measures can be taken accordingly.

Keywords

LDAP, User account management, Network security, Access control, Attack prevention, Audit

Índice de contenidos

Tabla de contenido

Índice de Figuras

Índice de Tablas

1. Introducción

1.1 Motivación y antecedentes

A la hora de gestionar el acceso de usuarios en una red determinada, resulta compleja y laboriosa la administración de cuentas locales almacenadas en los distintos equipos de la red. Para solventar este problema, se puede optar por centralizar las cuentas y la información de usuario en un servidor. Una implementación posible de esta solución es el uso de Lightweight Directory Access Protocol (LDAP). LDAP es un protocolo estándar que se implementa sobre la capa de aplicación de conexiones del modelo TCP/IP (Transmission Control Protocol/Internet Protocol) y define el acceso y representación de la información contenida en un directorio. Un servicio de directorio se puede definir como un conjunto de datos, esencialmente estático en el tiempo, optimizado para realizar consultas. Por ejemplo, la ya mencionada información de los usuarios de una red [1].

Pese a las ventajas que ofrece esta solución, pueden existir distintos problemas inherentes al protocolo que deben ser tenidos en cuenta a la hora de trabajar con él. Existen una gran cantidad de implementaciones diferentes del mismo, cada una con características y funcionalidades propias. Además, destaca su antigüedad, pues fue creado en la década de 1990 sobre el estándar X.500 [2] y desarrollado para hacer frente a las necesidades de la época, en especial, poder acceder a directorios de forma eficiente, algo que no lograba su predecesor Directory Access Protocol (DAP). Gran parte de la problemática del protocolo proviene de su misma creación. Al heredar gran parte de las características de X.500, nació como un protocolo complejo, característica que lo ha definido durante toda su existencia. Otro de los problemas consiste en que fue desarrollado sin tener en cuenta las técnicas y necesidades que existen hoy en día en seguridad informática, lo que lo puede hacer potencialmente inseguro si se quiere implementar en sistemas actuales [3]. Por lo tanto, para estar al día con las necesidades actuales en materia de seguridad y de adaptación a los sistemas presentes, se han requerido actualizaciones constantes y periódicas, por ejemplo, para añadir soporte con Transport Layer Security (TLS) [4], protocolo que ofrece comunicaciones de red seguras.

El uso de LDAP para centralizar la información de acceso a una red también conlleva otra serie de problemas e inconvenientes a tener en cuenta. Ni el protocolo ni sus diferentes implementaciones ofrecen un método sencillo y claro para monitorizar y clasificar la información relevante de los eventos producidos en la red, siendo necesario instalar complementos para obtener esta funcionalidad. En concreto, al utilizar LDAP en este caso para centralizar la información de acceso en la red, sería clave y esencial contar con datos acerca de los intentos de acceso que se producen en los diferentes equipos de esta. Estos complementos se han ido desarrollando junto con las distintas implementaciones de LDAP y han aportado seguridad y capacidad de monitoreo al protocolo. Por ejemplo, diversas implementaciones del protocolo han desarrollado extensiones que se pueden instalar sobre la implementación base para ofrecer diferentes funcionalidades, como registros de auditoría o control de acceso a la información almacenada. También se han desarrollado aplicaciones, como FusionDirectory o ApacheDS, que se instalan junto con las implementaciones LDAP, ampliando sus características y ofreciendo un manejo más cómodo y sencillo. Pese a estos posibles parches que complementan al protocolo, se sigue sin contar con un medio claro para recopilar y mostrar la información de los eventos producidos en la red a través de LDAP. Estas carencias en seguridad podrían implicar que información tan relevante como el intento de acceso de un usuario a un equipo para el que no tiene acceso no quede registrado y no sea mostrado al administrador para que pueda tomar acciones en respuesta. En consecuencia, el objetivo de este trabajo será desarrollar una solución, utilizando los medios que ofrece LDAP y las distintas extensiones y aplicaciones ya mencionadas para resolver esta carencia del protocolo en su uso como medio de centralización de la información de acceso a una red.

Para desarrollar la solución mencionada, se hará uso de los diferentes recursos que ofrecen las implementaciones del protocolo, con el objetivo de reunir información adicional acerca de los eventos ocurridos en el servidor que da acceso a LDAP. La implentación OpenLDAP será la solución escogida para el desarrollo gracias a ser de código abierto, ampliamente utilizada y flexible. Otra característica a favor de OpenLDAP es contar con extensiones, llamadas

overlays, que ofrecen funcionalidades extendidas. Varios de estos overlays, como AccessLog permiten registrar logs de las búsquedas realizadas en el servidor [1], aunque la información que se puede obtener de estos logs es limitada y resulta complicado determinar la información relevante para un posible lector. Algo similar ocurre con los propios logs con los que cuenta OpenLDAP, ya que ofrecen una gran cantidad de información y esta resulta poco legible, sobre todo si no se conocen en profundidad las diferentes modalidades que tiene esta implementación de guardar logs. Por último, se hará uso de las aplicaciones que se instalan junto con LDAP para facilitar su gestión, aunque cabe resaltar que estas también ofrecen una funcionalidad limitada en este aspecto. Distintas aplicaciones de uso extendido, como Softerra o FusionDirectory ofrecen un registro de información similar al de la implementación base de LDAP [5], o en el caso de incluir funcionalidades de auditoría adicionales, estos se limitan a monitorear y registrar los accesos a la aplicación, no al servidor LDAP al completo, algo que sucede por ejemplo con el plugin de auditoría de FusionDirectory [6].

Tomando estas herramientas como base se construirá la implementación que pueda registrar los accesos producidos en la red, haciendo énfasis en detectar accesos fraudulentos de usuarios a equipos para los que no tienen acceso. Se buscará mostrar esta información de forma clara y ordenada, ofreciendo al usuario la posibilidad de filtrar la información requerida. Gracias a ello, se podrán detectar acciones maliciosas o no permitidas y, por tanto, mejorar la seguridad en la red gracias a contar con información de calidad sobre los eventos producidos en ella.

1.2 Objetivos

El objetivo fundamental del proyecto es desarrollar una herramienta que permita detectar los accesos ocurridos en la red, mediante el uso del protocolo LDAP.

Para lograr este objetivo, se deberá realizar una serie de objetivos de menor envergadura. Entre estos objetivos destacan tres objetivos principales, divididos en submetas. Como primera meta se establecerá construir un servidor LDAP funcional (O1). Para alcanzar este objetivo, se realizará una investigación acerca de las diversas implementaciones existentes para desarrollar el servidor LDAP

(O1.1). Se compararán las implementaciones y aplicaciones de LDAP, dando prioridad a desarrollar sobre código abierto y soluciones estandarizadas. Cuando ya se haya escogido un modelo a instalar, se desarrollará una red que imite a un escenario real (O1.2). Sobre esa red se configurará el servidor, que tendrá capacidad de recibir consultas desde la totalidad de la red local desarrollada (O1.3). De este modo, los usuarios de la red podrán autenticarse en las diferentes máquinas haciendo uso del servidor (O1.4). A partir de este punto, se desarrollará una herramienta que permita reunir y mostrar la información relevante acerca de las conexiones realizadas a un servidor LDAP (O2), con el objetivo de que un administrador de sistemas pueda hacer uso de ella para detectar posibles ataques o comportamientos no permitidos dentro de la red. Para ello, se configurará el servidor LDAP para que registre la información relevante (O2.1). Se guardarán las distintas conexiones en un fichero mediante un script (O2.2) para dejar constancia de los hechos ocurridos y realizar labores de auditoría. Se extraerá la información más relevante de cada conexión ocurrida, como la dirección IP desde la que se ha establecido la conexión o el usuario que la ha realizado. Una vez diseñado el script y el formato de almacenamiento de datos, se configurará el sistema para que se guarde la información de auditoría periódicamente en el fichero de conexiones (O2.3). Finalmente, se creará una interfaz gráfica para interactuar con la información de forma más simple y visual (O3). Como primer paso dentro de este objetivo, se desarrollará una Application Programming Interface Representational State Transfer (API REST), es decir, una interfaz capaz de interactuar con un servicio que implementa la arquitectura REST [7], con el objetivo de interactuar con la información generada por el objetivo anterior (O3.1). A continuación, se construirá una interfaz visual para interactuar con los distintos servicios que ofrece la API REST (O3.2). Gracias a la interfaz, el usuario podrá hacer uso de distintos filtros y condiciones para mostrar la información más relevante que requiera en cada momento, por ejemplo, mostrando solo las conexiones erróneas o las realizadas desde un equipo concreto.

1.3 Plan de trabajo

Para alcanzar los objetivos planteados, se seguirá una metodología de trabajo ad hoc, dividida en ciclos de trabajo de entre dos y tres semanas de duración.

Cada ciclo de trabajo se centrará en el desarrollo de los objetivos planteados en el proyecto. Al inicio de cada ciclo de trabajo, se realizará una reunión de seguimiento para revisar el trabajo realizado en el ciclo anterior y establecer el objetivo a realizar para el ciclo siguiente. Tras la reunión, comenzará el desarrollo de un objetivo específico, durante el tiempo de duración del ciclo, documentando los pasos realizados y la información más relevante acerca del desarrollo del proyecto. El tiempo de desarrollo para cada objetivo específico ocupará un tiempo variable, aunque será aproximadamente de dos ciclos de trabajo. En la Figura 1.1 se muestra un esquema con las distintas acciones realizadas en un ciclo de trabajo.

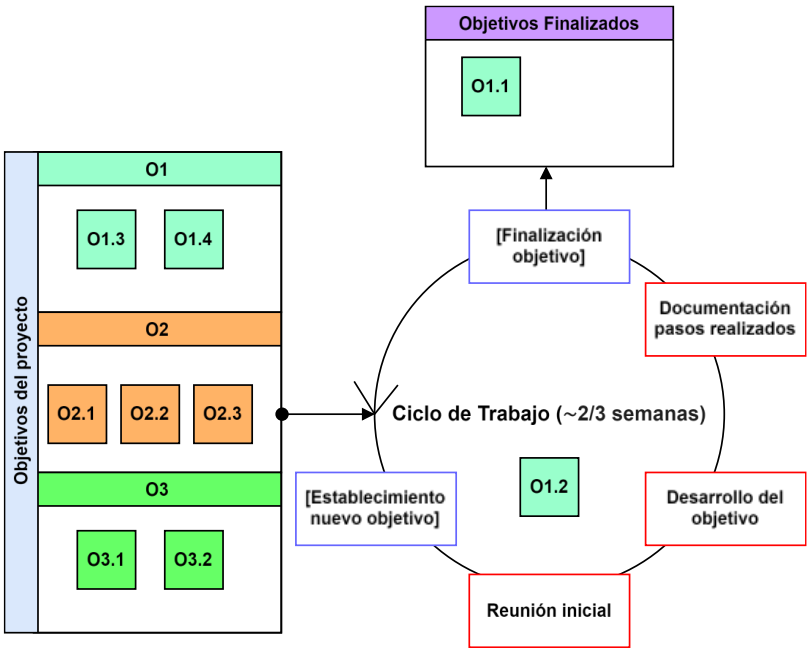


Figura 1.1 Diagrama de Ciclo de Trabajo

La metodología seguida recoge características de varias metodologías de desarrollo de software. En concreto, del desarrollo en cascada se obtiene la metodología lineal utilizada para implementar los objetivos. Estos se ejecutan uno por uno, siendo necesario la finalización completa de un objetivo antes de continuar con el siguiente. Por otra parte, de la metodología ágil SCRUM se obtienen los ciclos de trabajo utilizados en el proyecto, compartiendo algunas características con los sprints de la metodología mencionada. Algunas similaridades son la duración del ciclo y las reuniones realizadas al principio de

este para determinar el trabajo a realizar durante este y la revisión del trabajo ya finalizado. Sin embargo, al tratarse un proyecto unipersonal, no son necesarias las reuniones y procedimientos de coordinación de equipos presentes en este tipo de metodologías.

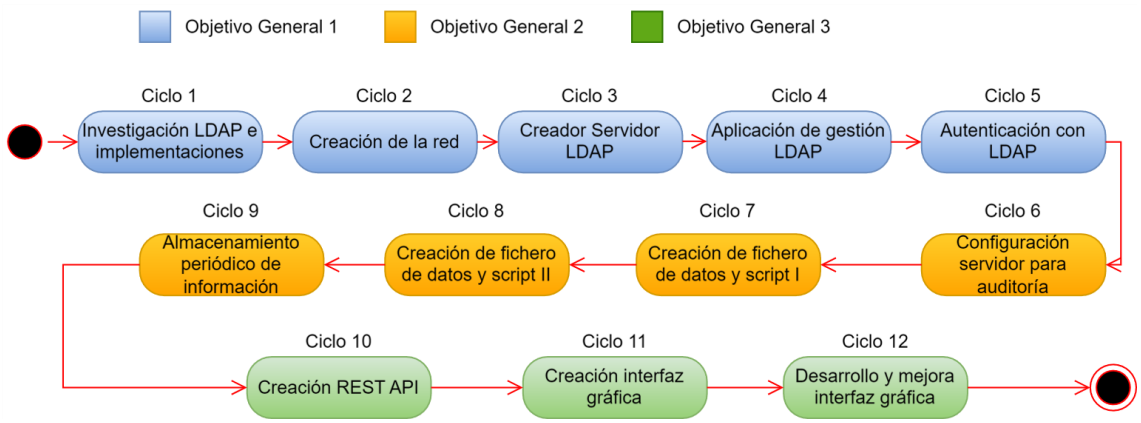


Figura 1.2 Flujo de ciclos de desarrollo del proyecto

En la Figura 1.2 se muestra un diagrama de flujo que representa el desarrollo del proyecto, clasificando los ciclos según el objetivo general desarrollado. Como se ha mencionado anteriormente, un ciclo durará entre dos y tres semanas y los objetivos específicos tienen una duración de ciclos variable, por lo que un objetivo específico podrá estar formado por uno o más ciclos.

1. Introduction

1.1 Motivation and background

When managing user access on a given network, the administration of local accounts stored on different computers in the network is complex and time-consuming. To overcome this problem, one option is to centralize accounts and user information on a server. One possible implementation of this solution is the use of Lightweight Directory Access Protocol (LDAP). LDAP is a standard protocol that is implemented on the application layer of the TCP/IP (Transmission Control Protocol/Internet Protocol) model. LDAP also defines the access and representation of the information contained in a directory. A directory service can be defined as a set of data, essentially static in time, optimized for queries. For example, the previously mentioned information on the users of a network [1].

Despite the advantages offered by this solution, there may be different problems inherent to the protocol that must be taken into account when working with it. There are a large number of different implementations of the protocol, each with its own characteristics and functionalities. In addition, its antiquity stands out, since it was created in the 1990s on the X.500 standard [2] and developed to meet the needs of the time, especially to be able to access directories efficiently, something that its predecessor Directory Access Protocol (DAP) could not achieve. A large part of the protocol's problems stems from its very creation. By inheriting many of the characteristics of X.500, it was born as a complex protocol, a characteristic that has defined it throughout its existence. Another problem is that it was developed without taking into account the techniques and needs that exist today in computer security, which can make it potentially insecure if it is implemented in current systems [3]. Therefore, in order to keep up with today's security needs and to adapt to current systems, constant and periodic updates have been required, for example, to add support for Transport Layer Security (TLS) [4], a protocol that provides secure network communications.

The use of LDAP to centralize network access information also brings with it a number of other problems and drawbacks to consider. Neither the protocol nor its different implementations offer a simple and clear method to monitor and

classify the relevant information of the events produced in the network, being necessary to install add-ons to obtain this functionality. In particular, when using LDAP in this case to centralize the access information in the network, it would be key and essential to have information about the access attempts that occur in the different computers of the network.

These add-ons have been developed along with the various LDAP implementations and have added security and monitoring capabilities to the protocol. For example, various implementations of the protocol have developed extensions that can be installed on top of the base implementation to provide different functionalities, such as audit logs or access control to stored information. Applications have also been developed, such as FusionDirectory or ApacheDS, which are installed together with LDAP implementations, extending their features and offering easier and more convenient management. Despite these possible patches that complement the protocol, there is still no clear means of collecting and displaying information on events occurring on the network via LDAP. These gaps in security could mean that information as relevant as a user's attempt to access a computer to which he does not have access is not recorded and is not displayed to the administrator so that he can take action in response. Consequently, the objective of this work will be to develop a solution, using the means offered by LDAP and the different extensions and applications already mentioned, to solve this shortcoming of the protocol in its use as a means of centralizing access information to a network.

To develop the aforementioned solution, we will make use of the different resources offered by the protocol implementations, with the objective of gathering additional information about the events occurring in the server that gives access to LDAP. The OpenLDAP implementation will be the solution chosen for the development because it is open source, widely used and flexible. Another feature in favor of OpenLDAP is the availability of extensions, called overlays, which offer extended functionalities. Several of these overlays, such as AccessLog, allow to record logs of the searches performed on the server [1], although the information that can be obtained from these logs is limited and it is difficult to determine the relevant information for a potential reader. Something similar happens with the OpenLDAP logs themselves, since they offer a large amount of information that

is not very readable, especially if the different modalities of this log storage implementation are not known in depth. Finally, use will be made of the applications that are installed together with LDAP to facilitate its management, although it should be noted that these also offer limited functionality in this aspect. Several applications in widespread use, such as Softerra or FusionDirectory offer similar information logging to the base LDAP implementation [5], or in the case of including additional auditing functionalities, these are limited to monitoring and logging accesses to the application, not to the entire LDAP server, something that happens for example with FusionDirectory's auditing plugin [6].

Using these tools as a basis, the implementation will be built to record the accesses produced in the network, with emphasis on detecting fraudulent accesses of users to equipment to which they do not have access. This information will be displayed in a clear and orderly manner, offering the user the possibility of filtering the required information. Thanks to this, it will be possible to detect malicious or unauthorized actions and, therefore, improve network security thanks to having quality information on the events produced on the network.

1.2 Goals

The main objective of the project is to develop a tool to detect the accesses occurred in the network, by using the LDAP protocol.

In order to achieve this objective, a series of smaller objectives will have to be carried out. Among these objectives there are three main objectives, divided into subgoals. The first goal is to build a functional LDAP server (O1). To achieve this goal, an investigation of the various existing implementations for developing the LDAP server (O1.1) will be accomplished. LDAP implementations and applications will be compared, giving priority to developing open source and standardized solutions. Once a model to be installed has been chosen, a network that imitates a real scenario will be developed (O1.2). The server will be configured on this network, which will have the capacity to receive queries from the entire local network developed (O1.3). In this way, network users will be able to authenticate themselves on the different machines using the server (O1.4). From this point, a tool will be developed to gather and display relevant information about the connections made to the LDAP server (O2), so that a system

administrator can use it to detect possible attacks or unauthorized behavior within the network. To do so, the LDAP server will be configured to record the relevant information (O2.1). The different connections will be saved in a file by running a script (O2.2) to record the events that have occurred and to carry out auditing tasks. The most relevant information will be extracted from each connection, such as the IP address from which the connection was established or the user who made it. Once the script and the data storage format have been designed, the system will be configured to save the audit information periodically in the connections file (O2.3). Finally, a graphical interface will be created to interact with the information in a simpler and more visual way (O3). As a first step within this objective, an Application Programming Interface Representational State Transfer (API REST) will be developed, that is, an interface capable of interacting with a service that implements the REST architecture [7], in order to interact with the information generated by the previous objective (O3.1). Next, a visual interface will be built to interact with the different services offered by the REST API (O3.2). Thanks to the interface, the user will be able to make use of different filters and conditions to display the most relevant information required at any given time, for example, showing only the wrong connections or those made from a specific computer.

1.3 Workplan

In order to achieve the proposed objectives, an ad hoc work methodology will be followed, divided into work cycles of between two and three weeks' duration. Each work cycle will focus on the development of the objectives set out in the project. At the beginning of each work cycle, a follow-up meeting will be held to review the work done in the previous cycle and establish the objective to be achieved for the next cycle. After the meeting, the development of a specific objective will begin, for the duration of the cycle, documenting the steps taken and the most relevant information about the development of the project. The development time for each specific objective will vary in time, but will be approximately two work cycles. Figure 1.1 shows a scheme with the different actions performed in a work cycle.

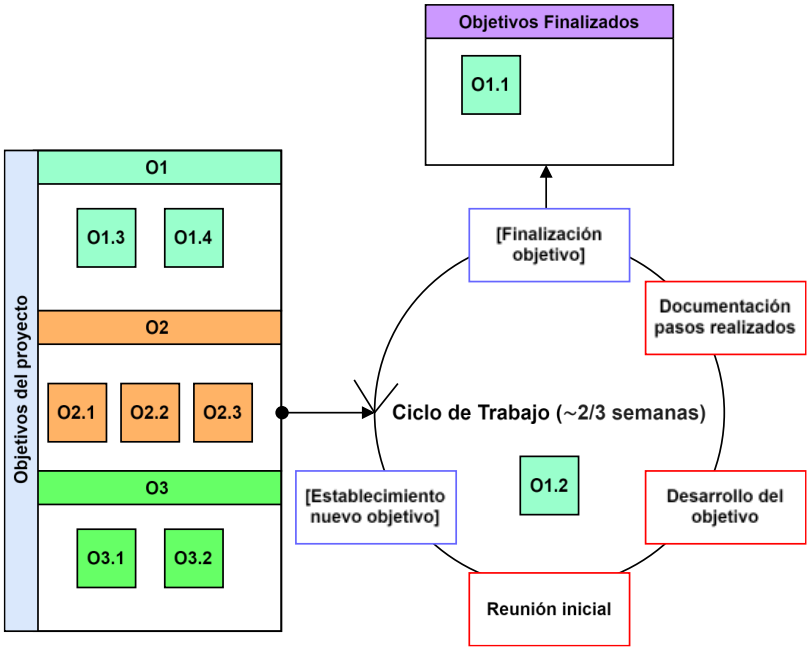


Figure 1.1 Work Cycle Diagram

The methodology followed gathers characteristics of several software development methodologies. Specifically, from the waterfall development we obtain the linear methodology used to implement the objectives. These are executed one by one, being necessary the complete completion of an objective before continuing with the next one. On the other hand, from the agile SCRUM methodology we obtain the work cycles used in the project, sharing some characteristics with the sprints of the mentioned methodology. Some similarities are the duration of the cycle and the meetings held at the beginning of the cycle to determine the work to be done during the cycle and the review of the work already completed. However, since it is a one-person project, the team coordination meetings and procedures present in this type of methodologies are not necessary.

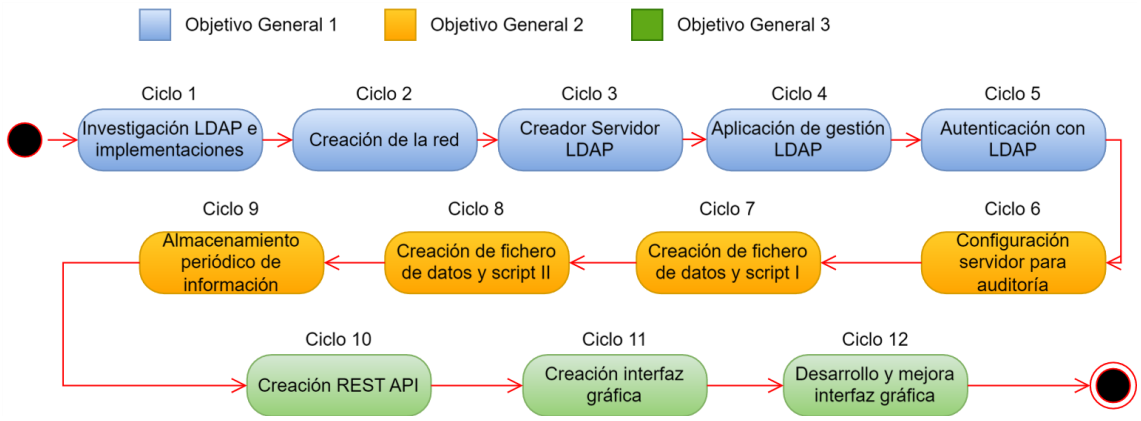


Figure 1.2 Project development cycle flow

Figure 1.2 shows a flow chart representing the development of the project, classifying the cycles according to the general objective developed. As mentioned above, a cycle will last between two and three weeks and the specific objectives have a variable cycle length, so that a specific objective may consist of one or more cycles.

2. El protocolo LDAP

2.1 Características del protocolo

LDAP es un protocolo construido sobre la pila TCP/IP que define el acceso y representación de la información de un directorio. Según la definición del estándar X.500, predecesor de LDAP, un directorio es una colección de sistemas abiertos cooperando para ofrecer servicios de directorio. El cliente que lo requiera accederá a la información del servidor que contenga el directorio a través de su usuario de directorio y utilizando un protocolo de acceso al mismo [8]. Primero fue creado DAP, compatible con el modelo Open Systems Interconnection (OSI) y más adelante, en la década de 1990, se desarrolló LDAP, compatible con TCP/IP, siendo este el protocolo de acceso a directorio que finalmente se estableció como estándar. Las distintas versiones y características del protocolo están definidas en las Request For Comments (RFC), documentación técnica de la organización Internet Engineering Task Force [9]. La versión actual del protocolo, LDAPv3, se encuentra definida en la RFC 4511.

LDAP es un protocolo que define el método de acceso a la información, la forma en que se organiza, las operaciones que se pueden realizar con la información y el modo en el que se protege la información de accesos indebidos [10]. Todas estas funcionalidades pueden ser también cubiertas por una base de datos relacional, por ejemplo, usando el lenguaje Structured Query Language (SQL). Pese a que LDAP y una base de datos relacional puedan cumplir las mismas funciones, existen diferencias entre ambas implementaciones, y diferentes casos de uso en el que puede ser óptimo utilizar LDAP o utilizar una base de datos, dependiendo de los datos a acceder y el modo en el que van a ser accedidos.

Para empezar, LDAP está pensado y optimizado para acceder a datos que son mayoritariamente estáticos. Es decir, que los datos almacenados cambien con poca frecuencia. Esto se debe a que LDAP está optimizado para realizar lecturas de la información almacenada y al tener que actualizar los índices en las modificaciones, el rendimiento se verá reducido. Otro elemento que distingue a LDAP es la abstracción que realiza sobre el almacenamiento físico de los datos, pudiendo estos estar almacenados en distintos formatos. Este último aspecto

confronta directamente con la idea de una base de datos relacional, ya que esta define perfectamente cómo debe ser almacenada la información [3]. Finalmente, para las bases de datos relacionales, no resulta óptimo el modelo de datos jerárquico que soporta LDAP. Este modelo, permite almacenar atributos multivalor, puede tener claves primarias complejas de determinar y atributos que se repiten en muchos objetos de datos. Todos estos aspectos dan lugar a estructuras tabulares ineficientes y redundantes, lo que daría lugar a un bajo rendimiento en caso de implementar una estructura jerárquica como las que se utilizan comúnmente en LDAP mediante una base de datos relacional [1]. En conclusión, el protocolo LDAP resulta óptimo para acceder a información que se modificará en pocas o ninguna ocasión. También resulta de útil aplicación a la hora de almacenar datos jerárquicos, poco eficientes de almacenar en bases de datos relacionales y si se requiere abstracción entre el modo de acceso a los datos y el modo físico de almacenarlos.

Otro aspecto relevante de LDAP es el modo en el que representa la información almacenada. La información se agrupa en entradas, que se corresponden con todas las entidades sobre las que queremos tener datos almacenados, como una división de la empresa, un empleado o un ordenador de trabajo. Cada entrada será representada por un objeto, que es una agrupación de distintos atributos. Estos objetos, denominados objectClasses*,* definen los atributos que deben poseer. Además, también definen si los atributos deben ser incluidos o son opcionales. Asimismo, los objectClasses pueden tener relaciones de herencia con otros objectClasses*.* Por ejemplo, un objeto como inetOrgPerson extiende al objeto organizationalPerson, poseyendo todos sus atributos y extendiendo el objeto con varios atributos adicionales, de forma similar a los objetos presentes en la programación orientada a objetos. Los atributos son elementos que almacenan datos de un tipo concreto. Tanto el tipo de dato que almacenará el atributo como la posibilidad de almacenar un solo valor o varios está especificado en la definición del atributo.

dn: uid=sergioa,ou=people,dc=redldap,dc=es

objectclass: inetOrgPerson

cn: Sergio

cn: Sergio Atienza

sn: Atienza

uid: sergioa

userpassword: 123

mail: sergioa@redldap.es

description: Trabaja en Host 2.1

ou: Recursos Humanos

Figura 2.1 Ejemplo de entrada del árbol LDAP

La entrada mostrada en la Figura 2.1 ejemplifica cómo se podría representar la información de una persona almacenada en LDAP. La persona pertenece al objectClass inetOrgPerson. Esta clase, en su definición, especifica que debe poseer una serie de atributos, los cuales son los que posee la persona “Sergio”, como cn (common Name), sn (surename), o uid (user IDentification).

Los atributos y los objectClasses son definidos globalmente, por lo que un objectClass o un atributo debe tener un nombre y un identificador de objeto, OID, únicos. Estos elementos son definidos en contenedores llamados schemas. Los schemas son agrupaciones de definiciones de objectClasses y atributos, en los que se indican todas las características que pueden poseer estos, como por ejemplo el tipo de datos del atributo los atributos que posee un objectClass. Es preferible, a la hora de diseñar la estructura de datos LDAP a implementar, utilizar schemas de uso estandarizado, para asegurar la interoperabilidad y compatibilidad entre sistemas. Por ejemplo, para definir una persona, la implementación OpenLDAP, incluye un schema de uso estándar llamado inetOrgPerson.schema [1], que incluye tanto la definición del objectClass como de sus atributos.

```
( 2.16.840.1.113730.3.1.3
    NAME 'employeeNumber'
    DESC 'numerically identifies an employee within an
organization'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
    SINGLE-VALUE )
```

Figura 2.2 Definición de atributo "employeeNumber" en RFC 2798

```
( 2.16.840.1.113730.3.2.2
   NAME 'inetOrgPerson'
   SUP organizationalPerson
   STRUCTURAL
   MAY (
      audio $ businessCategory $ carLicense $ departmentNumber $
      displayName $ employeeNumber $ employeeType $
givenName $
      homePhone $ homePostalAddress $ initials $ jpegPhoto $
      labeledURI $ mail $ manager $ mobile $ o $ pager $
      photo $ roomNumber $ secretary $ uid $ userCertificate $
      x500uniqueIdentifier $ preferredLanguage $
      userSMIMECertificate $ userPKCS12
   )
)
```

Figura 2.3 Definición de objectClass "'inetOrgPerson'" en RFC 2798

La definición del atributo mostrado en la Figura 2.2 y del objectClass mostrado en la Figura 2.3 son incluidas, junto con la definición de otros elementos en el schema mencionado, de uso estándar y descrito por la RFC 2798 [11]. En la

definición de ambos elementos, destaca el OID, el nombre global asignado y la posterior definición de las características del atributo, como la descripción y el número de valores aceptados en el caso del atributo o la herencia y los atributos admitidos en el caso del objectClass.

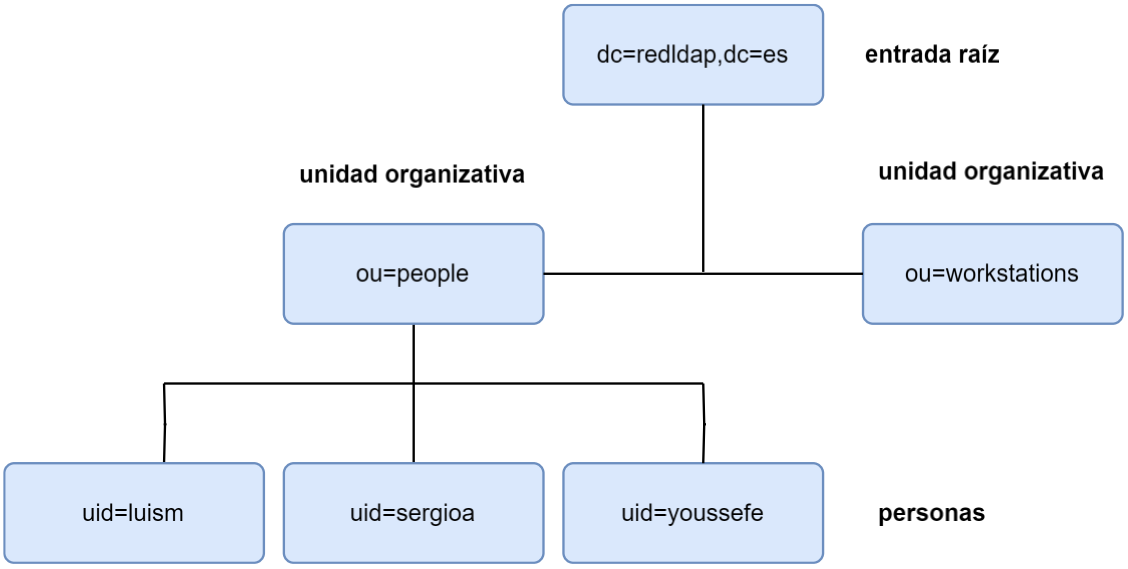


Figura 2.4 Ejemplo DIT LDAP

Todas estas entradas estarán organizadas de forma jerárquica, formando una estructura en forma de árbol conocida como Directory Information Tree (DIT). La estructura, representada en la Figura 2.4, permitirá identificar de forma única a cada entrada según la posición que ocupe en la jerarquía. Todas las entradas serán parte del dominio de una entrada superior, excepto la entrada raíz, de la cual heredará el resto de las entradas del árbol. Cada entrada obtendrá un nombre, conocido como Distinguished Name (DN), que la identificará de forma única en la jerarquía. Este nombre estará formado por la unión de los Relative Distinguished Names (RDN) de las entradas superiores. Este RDN será un atributo o combinación de atributos que posea un valor distintivo único para la entrada, como por ejemplo la uid de un usuario o su email.

```
dn: uid=sergioa,ou=people,dc=redldap,dc=es
```

Figura 2.5 Ejemplo de Distinguished Name

En la Figura 2.5 se puede observar que el atributo utilizado para definir el RDN es el uid de la persona “Sergio”. Esta entrada pertenecerá al dominio de ou=people, que almacenará las personas de la organización y este a su vez heredará de redldap.es, entrada raíz. Las entradas raíz, de forma estándar, según la RFC 2247 [12], se pueden formar a partir de un nombre de dominio como los utilizados en DNS (Domain Name System). En este caso, partiendo del nombre de dominio de la empresa ficticia, redldap.es, se formará el DN almacenando cada componente del nombre mediante un atributo dc, dando lugar a dc=redldap,dc=es.

LDAP ofrece varios métodos a los usuarios para interactuar con la información almacenada. Estos métodos son la búsqueda, la comparación, la actualización y la autenticación. La búsqueda realizará una consulta al servidor LDAP, para devolver las entradas coincidentes con los filtros de búsqueda especificados. Entre estos filtros pueden incluirse los valores de los atributos de la entrada a buscar, la entrada del árbol a partir de la cual buscar, según su distinguished name, o el número de entradas resultado a mostrar. Por otra parte, la comparación retornará si el atributo de una entrada posee el valor especificado. Las operaciones permitirán modificar, añadir, eliminar o renombrar entradas del DIT. Finalmente, la autenticación permitirá a un usuario identificarse en una sesión con el servidor. Esta autenticación se utilizará para determinar los permisos del usuario en el momento en que realiza las operaciones de búsqueda o actualización. El usuario autenticado podrá ser un usuario almacenado en LDAP, que se identificará mediante su nombre y contraseña, o un usuario anónimo, que no proveerá datos de identificación o proveerá el nombre y no la contraseña. Los permisos de los distintos usuarios y de usuarios anónimos serán determinados en LDAP y almacenados junto con el resto de información. Estos permisos indicarán las operaciones que podrá realizar el usuario y sobre qué entradas o atributos del DIT podrá realizar estas operaciones. Por ejemplo, se puede otorgar permiso a un usuario para modificar su propia información y no la de otros usuarios.

Un método estándar que ofrece LDAP para poder realizar las operaciones de actualización de entradas, poder importar y exportar entradas y además servir potencialmente de backend para el servidor es el formato LDAP Data

Interchange Files (LDIF). Este formato también permite representar en archivos de texto las distintas entradas de una estructura LDAP. Además, puede utilizarse en una operación de adición de entradas para representar la operación a añadir. De forma similar, puede ser utilizado en el resto de las operaciones de actualización. Por tanto, este formato puede ser utilizado para realizar modificaciones en la jerarquía del servidor LDAP, además de para representar esta de forma estándar. Por ejemplo, se utiliza este formato para representar la entrada de la persona “Sergio”, en la Figura 2.1.

2.2 Implementaciones y aplicaciones de LDAP

En cuanto a implementaciones del protocolo destacan OpenLDAP, ApacheDS y OpenDJ como opciones de código abierto. El primer caso está escrito en C, mientras que los otros dos lo están en Java, por lo que para su uso sería necesaria la instalación de una máquina virtual de Java. Las tres opciones están disponibles para Windows, Linux y MAC y mantienen soporte a día de hoy. Como elementos diferenciadores, OpenLDAP cuenta con una mayor dificultad a la hora de su configuración, al funcionar mediante comandos. Por otra parte, las otras dos implementaciones cuentan con interfaz gráfica para facilitar su uso. Además, ApacheDS cuenta con Apache Directory Studio, una aplicación adicional que facilita su gestión [13]. Las tres opciones son compatibles además con la instalación de otras aplicaciones que faciliten su gestión, y poseen una gran comunidad de usuarios y documentación. Otra opción, de pago y específica de Windows es Active Directory, opción muy utilizada pero descartada por no utilizar una máquina Windows y por no ser de código abierto.

LDAP y en especial, implementaciones como OpenLDAP, resultan complejas de gestionar y comprender debido a que son herramientas que funcionan mediante línea de comandos, sin ninguna clase de interfaz visual para hacer más intuitivo su uso. La dificultad que conlleva conocer los diferentes comandos, sumado a la posibilidad de corrupción de datos si no se utilizan estos correctamente, dan lugar a un manejo tosco y reservado a usuarios con amplio conocimiento y experiencia en el uso de este tipo de implementaciones. Para remediar esta dificultad en la gestión y entendimiento del protocolo, se han desarrollado aplicaciones de gestión, que se instalan junto con la implementación base de

LDAP. Estas aplicaciones, como Apache Directory Studio [13] o FusionDirectory [6], ofrecen una interfaz visual para poder visualizar de forma sencilla los elementos almacenados en LDAP, además de poder manejarlos y modificarlos de forma sencilla, sin necesidad de conocer a fondo el funcionamiento de los diferentes comandos del protocolo. Por otra parte, también ofrecen opciones para configurar el servidor de forma más sencilla, por ejemplo, añadiendo opciones de seguridad o logs adicionales. Finalmente, como ocurre en FusionDirectory, la instalación viene incluida con schemas propios de la aplicación, que pueden resultar útiles dependiendo de la solución concreta que se quiera alcanzar, especialmente si estos schemas incluyen objectClasses y atributos estandarizados.

3. LDAP y seguridad

A la hora de gestionar la seguridad en un entorno LDAP se debe conocer cómo el protocolo maneja los permisos asociados a los usuarios, además de los principales ataques.

Todas las operaciones realizadas en un servicio LDAP, serán aplicadas por un usuario cuya información está registrada en el DIT, o por un usuario anónimo en caso de que no se especifique información de usuario. Los usuarios cuya información está almacenada en el servidor, además de poder realizar las operaciones de consulta y modificación del árbol, accederán a sus equipos mediante consultas a través de la información de usuario residente en el DIT. Esta primera consulta se realizará con el usuario anónimo previamente mencionado, ya que el usuario todavía no ha completado la autenticación. La existencia de este usuario anónimo supone por sí misma una posible vulnerabilidad para el servidor, pues puede comprometerse la confidencialidad de los datos almacenados. Por defecto este usuario solo tendrá permisos para realizar esta acción de autenticación, pero se debe tener en cuenta el potencial riesgo de configurar sus permisos de forma incorrecta, pudiendo exponer la información almacenada a cualquier usuario no autenticado. De la misma forma, al estar toda la información de todos los usuarios guardada en el DIT, las contraseñas de acceso de los usuarios, que también forman parte de la información almacenada, pueden llegar a ser accedidas o modificadas por usuarios que no deberían contar con permisos para ello. Estos hechos serían especialmente graves en caso de que sea información de un administrador la que se vea comprometida, poniendo en peligro la integridad del servidor al completo. Para gestionar los permisos de una forma adecuada, se definen en LDAP las Access Control Lists (ACLs). Estas listas, almacenadas junto con el resto de información en el DIT, especifican las personas que tienen acceso a una determinada zona del árbol y las operaciones que pueden realizar. Las entradas o atributos del árbol se especifican en estas listas mediante su DN, pudiéndose especificar además si la regla de acceso se aplica a la entrada o a niveles jerárquicamente inferiores de la misma también. Los usuarios también son especificados en la ACL, pudiendo además gestionar los permisos de usuarios anónimos o el usuario que posea el atributo del cual se está verificando el

acceso, mediante las cláusulas *anonymous* y *self*, respectivamente. *Self* puede ser utilizado por ejemplo para permitir a un usuario ver y modificar la información de su propia entrada del árbol. Los usuarios pueden ser especificados de varias formas, por ejemplo, dando permisos a todos los usuarios autenticados, a usuarios concretos o a usuarios de una parte del dominio específica. Esta última opción puede utilizarse para gestionar los permisos de usuario mediante grupos, otorgando diferentes permisos a los grupos de usuarios. Finalmente, en la ACL se concretarán los permisos concedidos a los usuarios. Estos permisos pueden ser none (ningún permiso), disclose (recibir información de error al realizar operaciones), autenticación, comparación, búsqueda, lectura, escritura y gestión. Cada nivel de permisos contendrá todos los permisos anteriores. Este formato de ACL, permitirá gestionar adecuadamente el permiso de los usuarios al realizar las distintas operaciones que LDAP soporta [14].

```
access to attrs=userPassword
    by self write
    by anonymous auth
    by dn.base="cn=Sergio,dc=redldap,dc=es" write
    by * none
```

Figura 3.1 Ejemplo ACL sobre atributo userPassword

En las ACLs mostradas en la Figura 3.1, se muestran los permisos de acceso para el atributo contraseña y para el resto del DIT. En el caso de la contraseña, se permite al usuario que posea dicha contraseña modificarla, a los usuarios anónimos se les permite acceder a ella para autenticarse solamente, al resto de atributos, al usuario Sergio se le otorga permiso de modificación también y al resto de usuarios (*) ningún permiso. Es importante resaltar que el usuario root de LDAP, rootdn, tendrá todos los permisos de lectura y escritura sobre todas las entradas. También es importante destacar que las ACLs se pueden asignar de forma global o para un back-end concreto, en caso de disponer de varios. Se aplicarán los permisos del back-end específico para las entradas almacenadas

en él, y en caso de no existir o no ser aplicable la ACL, se aplicarán las ACL globales [15]. Para asegurar de forma efectiva la confidencialidad e integridad de los datos almacenados, es recomendable adoptar una política de mínimo privilegio, es decir, limitar los permisos de usuario a las entradas y atributos estrictamente necesarias, asignándoles el mínimo privilegio requerido para que puedan realizar sus funciones. Se debe tener en cuenta, para no vulnerar la confidencialidad de datos, que el permiso por defecto si no existe una ACL es el de permitir la lectura a todos los usuarios. Finalmente, se debe tener especial precaución con los permisos asignados a usuarios anónimos, pues son una fuerte vulnerabilidad del sistema en caso de poseer permisos de acceso inadecuados al DIT.

Teniendo en cuenta esta configuración inicial básica, se puede proteger el servidor de los indebidos accesos de los usuarios registrados. Pese a la buena protección que pueden producir estas acciones, el servidor aún puede tener una serie de vulnerabilidades importantes, derivadas del propio funcionamiento de LDAP y de situarse en una red con acceso al exterior. Ambas casuísticas pueden tener contramedidas que reduzcan el riesgo, protegiendo el servidor y asegurando su confidencialidad, integridad de datos y disponibilidad de los usuarios a la hora de acceder a él.

Para comenzar, el servidor puede sufrir ataques debido a las conexiones en red que realiza el mismo. Estos ataques son comunes a los que podría recibir otra clase de servidor, y en el caso de un servidor de LDAP, existen algunas soluciones específicas para abordarlos.

3.1 Captura de paquetes

Uno de los principales ataques a sistemas de red es la interceptación de paquetes. Cuando un intruso se consigue infiltrar en la red y capturar los diferentes paquetes de una conexión, podrá leer los mismos si estos se transmiten como texto plano. LDAP no es ajeno a este problema. Cuando un cliente intenta autenticarse frente a un servidor, si la conexión no es segura, enviará toda su información de autenticación, es decir, su usuario y contraseña, como texto plano. Si un tercero puede leer esta información, la seguridad de la cuenta habría quedado completamente rota, como se muestra en la Figura 3.2.

Como solución, se puede configurar LDAP para establecer solo conexiones seguras, mediante TLS. Por defecto, LDAP escucha en el puerto 389 para conexiones estándar y en el 636 para conexiones LDAP sobre TLS. El encapsulado del protocolo LDAP sobre TLS se denomina LDAPS. Esta configuración básica se puede variar, estableciendo que el servidor acepte solo conexiones seguras mediante TLS. De este modo, se pueden evitar los ataques de escucha, al estar la conexión completamente cifrada, gracias a las funcionalidades de seguridad que ofrece TLS [3].

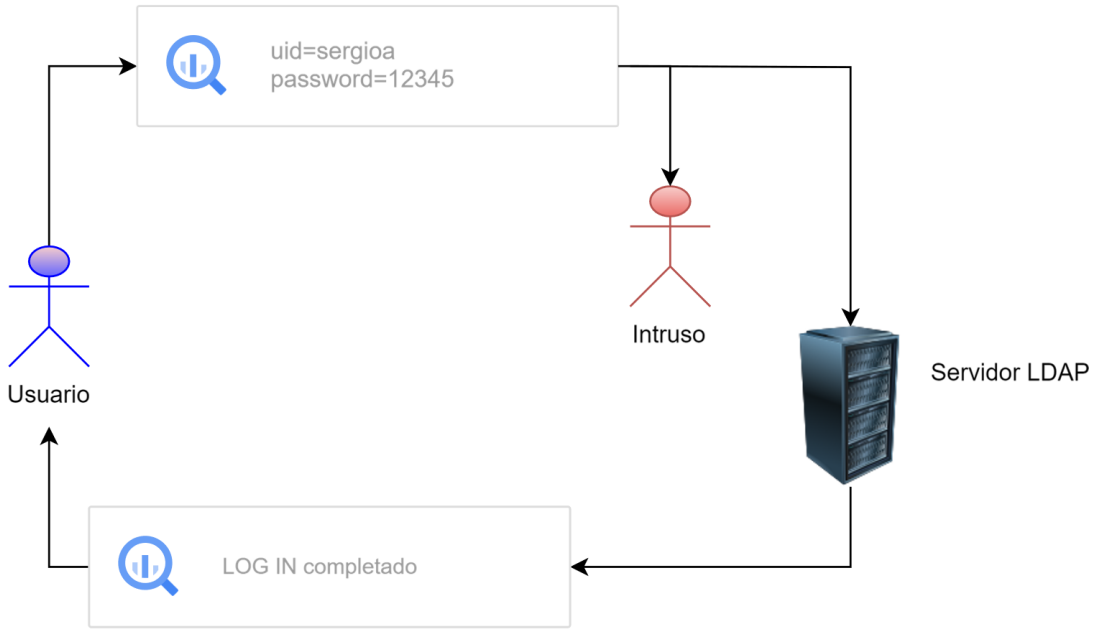


Figura 3.2 Captura de paquetes

3.2 Acceso e interceptación de contraseñas

Otro posible ataque que puede sufrir el servidor es la captura de las contraseñas, tanto en los archivos en los que se almacena localmente el DIT, como en las conexiones en las que el servidor envía una contraseña. En el primer caso, los archivos estarían protegidos mediante los permisos y sistemas de seguridad propios del sistema en el que se aloja el servidor, por lo que una adecuada seguridad del mismo puede mitigar y prevenir estos ataques. El segundo caso ocurre cuando se captura un paquete enviado por el servidor que contenga el atributo userPasswd, teniendo en cuenta que no se usa una conexión segura. Afortunadamente, la contraseña no se almacena ni se envía como texto plano,

sino que se realiza una función resumen, hash, de la misma y este valor es el que se guarda o se manda a través de la conexión establecida. Por tanto, la vulnerabilidad radica en el algoritmo utilizado para el hash. Si el algoritmo utilizado no es robusto, un atacante que robe las contraseñas podrá realizar un ataque de diccionario o utilizar rainbow tables[1] para conseguir averiguar el texto que origina los hashes obtenidos.

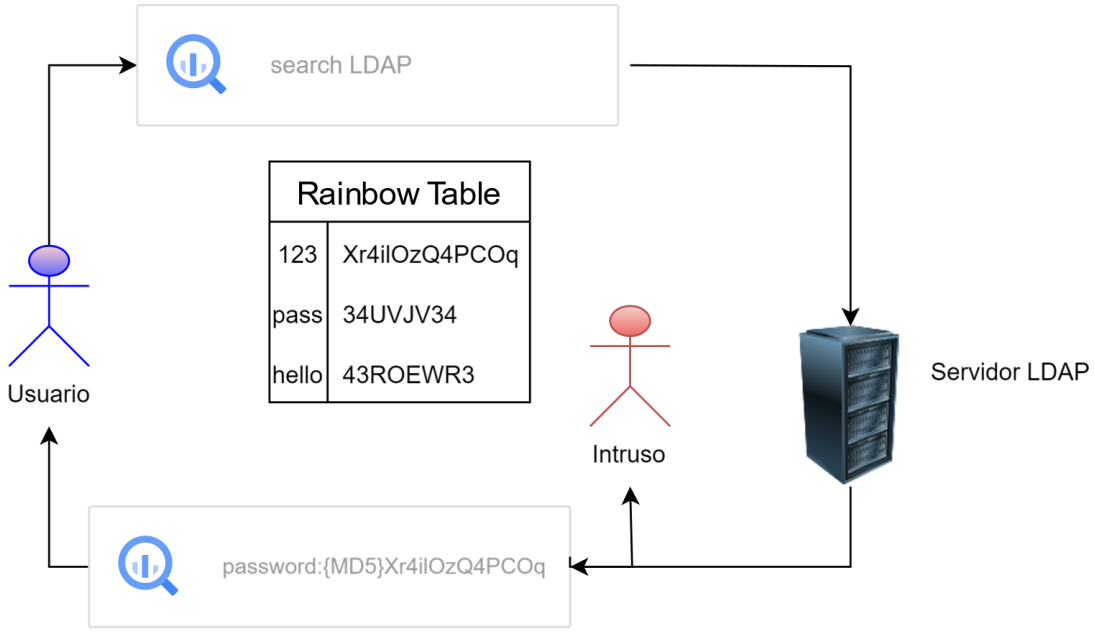


Figura 3.3 Ataque con Rainbow Table

En la Figura 3.3 se puede ver un ejemplo del ataque mencionado. Un intruso captura el hash de la contraseña, que está almacenado con un algoritmo inseguro, MD5, por lo que puede realizar varias acciones para descifrar el mismo. Puede optar por realizar un ataque de diccionario, probando con el hash de diferentes palabras, en especial, palabras comunes utilizadas en contraseñas. También, como se puede ver en la Figura 3.3, puede probar con rainbow tables, tablas con hashes ya realizados. En ambos casos, el uso de algoritmos de hash frágiles resulta en una vulnerabilidad, por la posibilidad de producirse colisiones. En el caso mostrado se utiliza el algoritmo MD5, que es un algoritmo inseguro por este motivo [16]. Para evitar las colisiones se deben utilizar solo algoritmos

[1] Tabla que contiene hashes de contraseñas comunes. Se utilizan para descifrar contraseñas almacenadas como hash.

seguros de hash. Dependiendo de la implementación de LDAP, se podrán utilizar unos u otros. Por ejemplo, una implementación segura con OpenLDAP, consiste en utilizar el algoritmo SHA con salt. Salt es un número aleatorio añadido al hash para aportarle una mayor seguridad. Otras acciones relevantes que se pueden adoptar para proteger las contraseñas son establecer requisitos a la hora de su complejidad y renovación. Ciertas implementaciones de LDAP permiten imponer estas condiciones, como el overlay ppolicy de OpenLDAP o las aplicaciones de gestión de LDAP, como FusionDirectory.

3.3 Denegación de servicio

Otros ataques distintos, que atentan directamente contra la disponibilidad del servicio LDAP son los ataques de denegación de servicio o denegación de servicio distribuido (DOS o DDOS). Estos ataques consisten en la saturación del servidor mediante la realización de una gran cantidad de conexiones simultáneas.

Figura 3.4 Ataque DOS

Para evitar estos ataques, ejemplificados con la Figura 3.4, se deberían tomar una serie de acciones básicas de seguridad en redes. Para comenzar, se debe limitar al máximo el contacto del servidor con el exterior, situando al mismo en una zona segura de la red. Un firewall será el encargado intermediar las conexiones entre el servidor y los clientes, impidiendo conexiones sospechosas y potenciales ataques. Otros mecanismos que ayudan a defenderse de ciertos ataques de denegación de servicio como TCP SYN flooding, es SYN cookies.

Este ataque satura el servidor mediante peticiones TCP SYN, y el mecanismo mencionado impide la saturación al iniciar la conexión solo cuando se recibe el ACK posterior [17]. Pese a estas herramientas, los ataques de denegación de servicio pueden suceder. Para prevenirlos y hacer el servidor más robusto frente a ellos, se puede tomar otra serie de acciones. Para ofrecer una mayor capacidad, resistencia a errores y robustez frente a ataques, LDAP soporta replicación y referenciación. Ambas opciones pueden ser desarrolladas de diversas formas, dependiendo de la funcionalidad concreta que se quiera conseguir y de la implementación LDAP escogida. En general, la replicación consiste en el uso de varios servidores con el mismo DIT, lo que permite tener servidores de respaldo si uno se satura o deja de funcionar, además de poseer varias copias de respaldo de los datos y ofrecer balanceo de carga por razones de rendimiento [3]. Por otra parte, la referenciación consiste en subdelegar el DIT en varios servidores. De forma similar a DNS, si se envía una petición a un servidor cuya respuesta no esté contenida en su porción del DIT, se recibirá una referencia del servidor que contenga la porción del DIT de la petición enviada [1], [3]. Esto permite gestionar por separado los fragmentos del DIT y ofrece robustez frente a caídas del servicio. Estas soluciones pueden hacer del servicio LDAP un servicio más robusto y proteger su disponibilidad frente a ataques externos.

3.4 LDAP Injection y LDAP Blind Injection

Otros ataques que puede sufrir el servidor, específicos del protocolo LDAP son LDAP injection y Blind LDAP injection. Estos ataques se producen debido a la forma en la que LDAP procesa los filtros en las búsquedas. Al introducir un usuario datos con un formato muy determinado se puede manipular la forma en el que el servidor procesa la petición, ganando acceso al DIT de forma ilegítima. Estos ataques funcionan de forma similar al conocido ataque SQL injection, pero son potencialmente mucho menos peligrosos que este. Son menos peligrosos debido a que los ataques concretos de LDAP solo se pueden realizar en búsquedas, no en modificaciones del árbol. Por lo tanto, solo pueden ser utilizados para autenticarse u obtener información del DIT, nunca para alterarlo, a diferencia de SQL injection, que puede ser utilizado también para eliminar y modificar tablas enteras en las bases de datos relacionales.

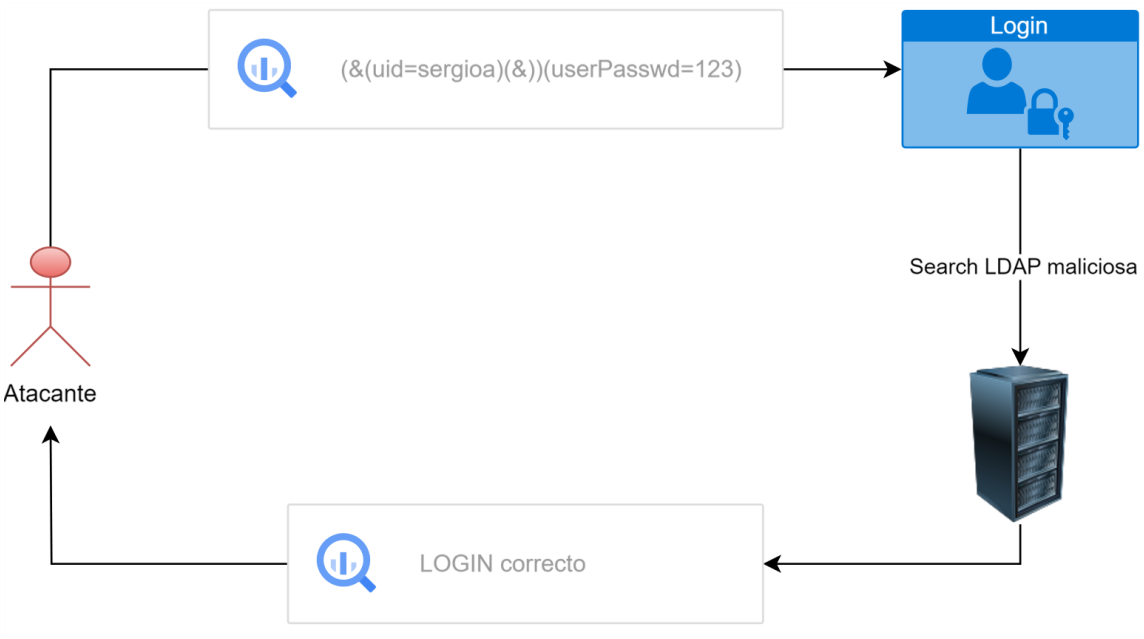


Figura 3.5 LDAP injection

LDAP injection puede ser utilizado para obtener privilegios o información mediante la manipulación de los filtros de búsqueda. Como se observa en la Figura 3.5, el atacante aprovecha los filtros de búsqueda para poder iniciar sesión sin saber la contraseña. Los filtros de búsqueda utilizados para el inicio de sesión serían (&(uid=id)(userPasswd=password)). El operador '&' indica que ambas condiciones deben ser cumplidas para pasar el filtro. En este caso, al introducir en el campo de formulario de nombre la cadena "sergioa)(&))", se construirá el filtro (&(uid=sergioa)(&))(userPasswd=123). El primer filtro devolverá "true", mientras que el segundo filtro, el de la contraseña, es ignorado. De este modo, un intruso ha podido introducirse en el sistema, solo conociendo el nombre de un usuario y el modo en el que el servidor procesa el filtro con el operador '&'. También puede darse el caso de que el filtro construido sea '|', es decir, con un operador condicional "or". De la misma forma, estos filtros también pueden ser explotados para obtener información del DIT. Un caso particular de LDAP injection es el ataque Blind LDAP injection, en el que el atacante puede inferir el resultado que devolverá una consulta. Por ejemplo, a la hora de buscar un usuario en la unidad organizacional "recursos humanos", se construirá el siguiente filtro: (&(uid=sergioa)(ou=recursos humanos)). Esta consulta devolverá todos los usuarios del área mencionada. Al introducir en el formulario de

búsqueda filtros adicionales, se podrá obtener una respuesta “true” o “false” del servidor, pudiendo así averiguar progresivamente información del servidor. Por ejemplo, al introducir “sergioa)(dirección=a*)” se formará el siguiente filtro: (&(uid=sergioa)(dirección=a*))(ou=recursos humanos)), que devolverá “true” si la dirección empieza por “a” o “false” en caso contrario. Realizando búsquedas sucesivas, el atacante podrá obtener los datos del empleado al completo. Estos dos ataques explotan la vulnerabilidad del servidor al procesar los datos introducidos por el usuario, debido al uso de los caracteres especiales que se utilizan en la construcción de los filtros (“(“, “)”, “&” y “|”) [18]. A la hora de realizar formularios para que el usuario introduzca datos, es imprescindible escapar los caracteres especiales, para evitar este tipo de ataques. Otras acciones adicionales compatibles con la anterior son realizar consultas parametrizadas y seguir la ya mencionada política de mínimo privilegio de usuario, para dificultar la realización de actividades maliciosas.

3.5 Accesos indebidos y actividad potencialmente maliciosa

Finalmente, la utilización de LDAP para centralizar la información de los usuarios de la red, con el objetivo de que estos se autentiquen mediante esta información, conlleva otra serie de posibles vulnerabilidades y carencias en seguridad. Un usuario podría intentar burlar la seguridad establecida en la red tratando de iniciar sesión en un ordenador ajeno al suyo. De la misma forma, un intruso podría intentar realizar ataques de fuerza bruta para averiguar la contraseña de un usuario de la red. Otros comportamientos sospechosos que podrían suponer un potencial ataque son conexiones a horas sospechosas y conexiones desde IPs no habituales. Las implementaciones estudiadas no ofrecen soluciones sencillas para detectar y auditar todos estos eventos. Pese a que LDAP tiene sus propios logs y sus diferentes implementaciones y aplicaciones ofrecen diferentes soluciones que permiten solventar alguno de estos problemas, no existe una solución que resuelva todo ello de forma clara y unificada. Por tanto, para poder detectar todos estos eventos y ofrecer a los administradores una solución que le permita detectar todos estos potenciales ataques o comportamientos indebidos, se desarrollará la implementación descrita en el capítulo 4.

4. Condiciones del entorno de desarrollo

4.1 Topología y características de la red

Para desarrollar la herramienta de monitoreo de LDAP que detecte los eventos mencionados, se creará una red de máquinas virtuales que simulará un escenario real. Este escenario podría corresponderse con una red corporativa, con los equipos de los empleados siendo los clientes del servidor LDAP y asignándole a este último la responsabilidad de almacenar la información de los distintos trabajadores. Los empleados realizarían consultas a este servidor para acceder a sus diferentes puestos de trabajo.

Los equipos contarán con la última versión de la distribución Debian de Linux, es decir, Debian 12. Esto se aplica a todos los equipos de la red excepto el servidor LDAP, que contará con Debian 11 debido a la falta de compatibilidad de algunas aplicaciones como FusionDirectory con la versión más reciente.

La red contará con los siguientes elementos:

1. Equipos que funcionarán como clientes del servidor LDAP, conectados a una red interna.
2. Equipo que funciona como servidor LDAP. Este servidor se encuentra en una red diferenciada para poder protegerlo mediante un firewall.
3. Equipo que funciona como encaminador, conectado a la red interna de clientes, a la del servidor LDAP y a Internet. Los distintos equipos de la red podrán acceder a Internet a través de esta máquina.
4. Servicio DHCP a través del encaminador. Esta máquina proveerá de direcciones IP dinámicas a los equipos clientes.
5. Firewall situado en el encaminador, implementado con la herramienta iptables. Este firewall enmascarará las direcciones IP de los equipos clientes al acceder al exterior, con NAT. También puede aplicar filtros para proteger al servidor LDAP de conexiones internas y externas.
6. Servicio DNS ofrecido por el encaminador. De esta forma, se podrán resolver los nombres asignados a los equipos de la red.

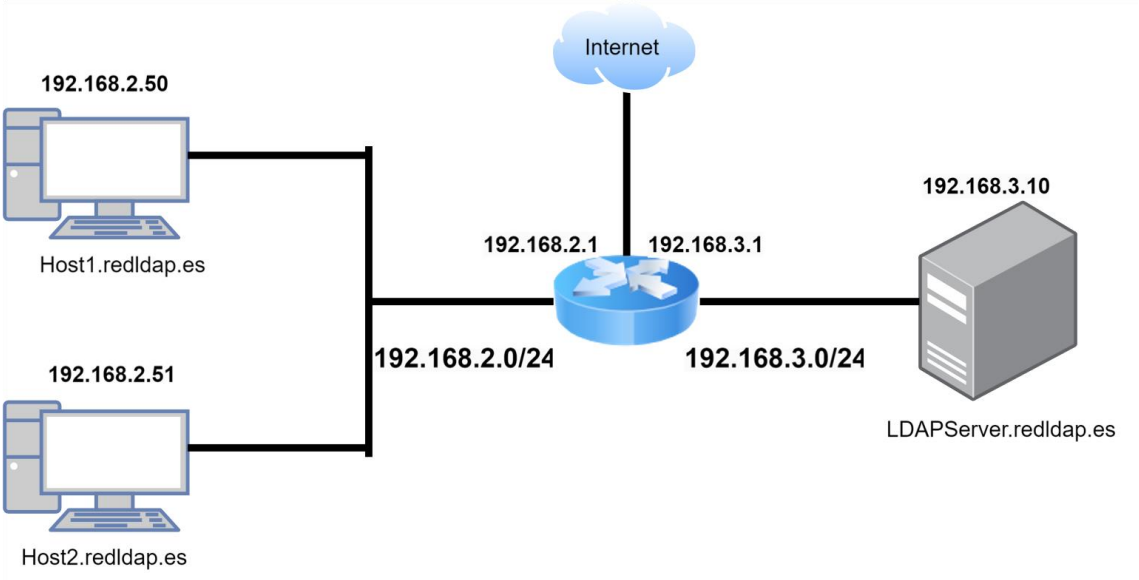


Figura 4.1 Estructura de Red

La topología descrita se representa mediante la Figura 4.1, en la que se indica la IP de cada red y las IP asignadas a cada máquina, de forma dinámica en el caso de los equipos cliente y con asignación estática en el resto de los casos. Por otra parte, se puede observar también el nombre de cada equipo y del dominio creado. El dominio de toda la red será redldap.es, que se corresponderá con el DN del dominio LDAP, que será cn=redldap, cn=es. Los nombres de los equipos serán usuarios numerados en inglés para facilitar su identificación en esta red ficticia. Para facilitar el ejemplo, cada equipo será accedido por un usuario que tenga el mismo número que él. Por ejemplo, se creará un usuario User One que será el único, además del administrador, con permisos de acceso al equipo cliente Host1.redldap.es. El usuario administrador, de nombre “Sergio”, tendrá pleno acceso a todos los equipos de la red.

4.2 Configuración del servidor

Una vez creada la estructura de la red e instalados los servicios necesarios para el funcionamiento de esta, se configurará el servidor LDAP. Para poner en marcha este servicio existen diversas implementaciones tanto de pago como de código abierto, por lo que se deberá escoger la más adecuada según las características deseadas del servidor a desarrollar y las necesidades de los

posibles clientes. Pese a la mayor facilidad de uso de ApacheDS y OpenDJ, además de las buenas prestaciones y soporte que ofrecen, se optará por implementar el servidor con OpenLDAP. El motivo es que esta opción está muy estandarizada, siendo parte de varias distribuciones de Linux como Ubuntu, es muy flexible y existe gran cantidad de documentación sobre ella [19].

Tras instalar OpenLDAP en el servidor, se podrá hacer uso de dos herramientas para manejar el servidor, ambas mediante línea de comandos. La primera es slapd, que es el demonio que funciona como servidor de LDAP [20]. Este demonio contará con una serie de comandos, comenzados por "slapd", que servirán para interactuar con él. Por ejemplo, slapadd servirá para añadir entradas al servidor. Estos comandos no realizan las modificaciones mediante las operaciones del protocolo LDAP, sino que modifican directamente la base de datos del servidor. Su uso con el servidor activo puede dar lugar a la corrupción del mismo, por lo que se recomienda tener el servidor parado a la hora de ejecutar estos comandos [21]. La otra herramienta instalada es el paquete ldap-utils, que proveerá comandos para interactuar con el servicio LDAP. Estos comandos sí utilizarán las operaciones descritas en el capítulo 2, como la búsqueda o la actualización, para interactuar con el servidor. Por ejemplo, el comando ldapadd realizará una operación de actualización para añadir entradas o ldapsearch la realizará de consulta para realizar búsquedas LDAP [22]. Con estos comandos no existe el riesgo de corrupción de datos que sí ocurre con los comandos de slapd [3]. OpenLDAP posee dos posibles formas de configurar el servidor. La primera es mediante configuración estática. En este modo, se configurará el servidor mediante un archivo de texto, "ldap.conf". Esta opción ha sido marcada como obsoleta y se recomienda no utilizarla. La opción restante, más moderna, es la configuración dinámica. Esta alternativa consiste en guardar la configuración del servidor en distintos archivos en formato LDIF. De esta manera el servidor podrá ser modificado en tiempo de ejecución mediante los comandos mencionados anteriormente, sin necesidad de parar el servidor para ello, algo que ocurre con la configuración estática [1].

```
dn: uid=userone,ou=people,dc=redldap,dc=es

objectclass: inetOrgPerson
```

cn: User

cn: User One

sn: One

uid: userone

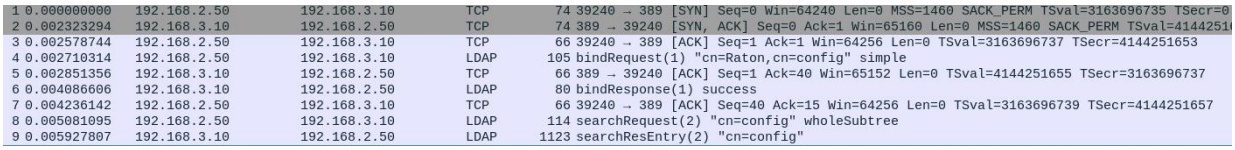
userpassword: one

mail: userone@redldap.es

description: Trabaja en Host1

ou: Recursos Humanos

Figura 4.2 Ejemplo de LDIF para añadir entrada al DIT



1	0.000000000	192.168.2.50	192.168.3.10	TCP	74	39240 → 389 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3163696735 TSecr=0
2	0.002323294	192.168.3.10	192.168.2.50	TCP	74	389 → 39240 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=4144251
3	0.002578744	192.168.2.50	192.168.3.10	TCP	66	39240 → 389 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3163696737 TSecr=4144251653
4	0.002710314	192.168.2.50	192.168.3.10	LDAP	105	bindRequest(1) "cn=Raton,cn=config" simple
5	0.002851356	192.168.3.10	192.168.2.50	TCP	66	389 → 39240 [ACK] Seq=1 Ack=40 Win=65152 Len=0 TSval=4144251655 TSecr=3163696737
6	0.004086606	192.168.3.10	192.168.2.50	LDAP	80	bindResponse(1) success
7	0.004236142	192.168.2.50	192.168.3.10	TCP	66	39240 → 389 [ACK] Seq=40 Ack=15 Win=64256 Len=0 TSval=3163696739 TSecr=4144251657
8	0.005081095	192.168.2.50	192.168.3.10	LDAP	114	searchRequest(2) "cn=config" wholeSubtree
9	0.005927807	192.168.3.10	192.168.2.50	LDAP	1123	searchResEntry(2) "cn=config"

Figura 4.3 Captura Wireshark conexión LDAP

Con el servidor en funcionamiento, se podrá comenzar a construir el DIT, a través del comando ldapadd y archivos en formato LDIF como el mostrado en la Figura 4.2. En este caso, se añade a la persona userone, usuario del equipo Host1.redldap.es. Por otra parte, también se podrán realizar consultas al servidor desde las distintas máquinas que componen la red, con ldapsearch. En la Figura 4.3 se muestra en Wireshark un ejemplo de una conexión de búsqueda LDAP. En concreto, se está consultando por cn=config, que es la sección del árbol que contiene la configuración del servidor en la configuración dinámica.

4.3 Aplicación de gestión LDAP y control de acceso

Como se observa, construir y gestionar mediante comandos el servidor LDAP puede ser complejo y dar lugar a fallos humanos, por la dificultad de uso de los comandos y por la posible corrupción de datos derivada del uso de estos. Para

poder desarrollar el servidor LDAP de una forma más sencilla y visual, se instalará una aplicación de gestión junto con la implementación OpenLDAP. De esta forma, se podrá construir el DIT, modificarlo y alterar la configuración del servidor de manera sencilla, sin tener que utilizar comandos y con una interfaz visual que facilita el trabajo. Además, mediante las aplicaciones se pueden instalar extensiones que añadan funcionalidades adicionales, útiles para mejorar la seguridad del servidor. En este tipo de aplicaciones, destacan algunas como Apache Directory Studio [13] o GoSA-project [23]. La primera aplicación es utilizable en OpenLDAP, aunque está especialmente pensado para ser utilizado junto con la implementación ApacheDS. Gosa-project por otro lado, no está diseñado para una aplicación en concreto. Ofrece una gran cantidad de funcionalidades para administrar un servidor LDAP y es de código abierto, pero actualmente no se continúa su desarrollo de forma oficial. Otro proyecto, surgido como un fork y una continuación de GoSA, también de código abierto y que ofrece una manera fácil y visual de trabajar con LDAP es FusionDirectory [6]. Este proyecto, que continúa oficialmente su desarrollo a día de hoy, ofrece una interfaz visual para gestionar LDAP, una serie de extensiones para añadir funcionalidades adicionales al servidor y posee esquemas estandarizados para poder interactuar con otras aplicaciones y servicios de forma sencilla. Existen otra serie de opciones de pago que ofrecen además servicios de soporte, como Softerra [5], en caso de querer optar por opciones que no son de código abierto.

Tras instalar FusionDirectory, se puede configurar el servidor y las distintas opciones de seguridad del mismo a través de esta aplicación. Por ejemplo, el algoritmo de hash en el que se almacenan las contraseñas o los puertos en los que escuchará el servidor. De este modo también se construirá el DIT, creando los usuarios con permiso de acceso a cada equipo. Para ello, se instalarán dos plugins: Portable Operating System Interface for UNIX (POSIX), para añadir atributos estándar a las cuentas de usuario y Systems, para añadir los equipos al árbol LDAP.

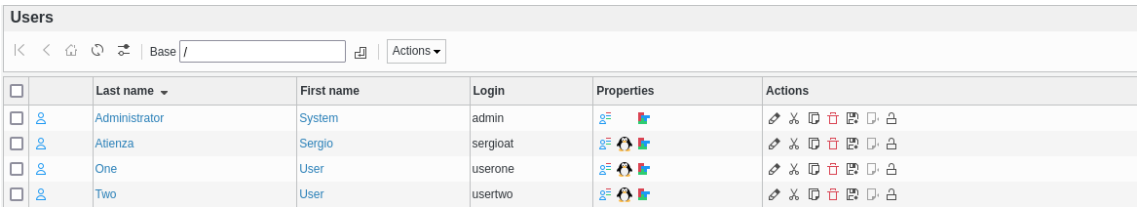


Figura 4.4 Usuarios en FusionDirectory

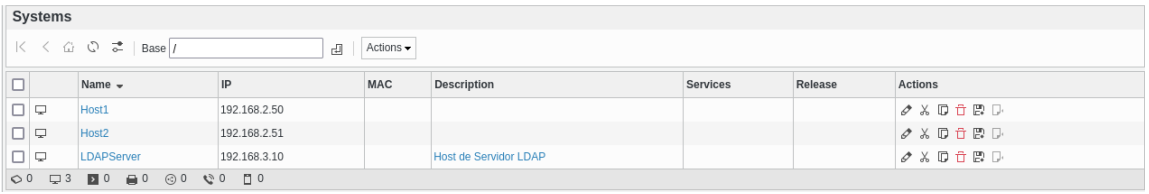


Figura 4.5 Sistemas en FusionDirectory

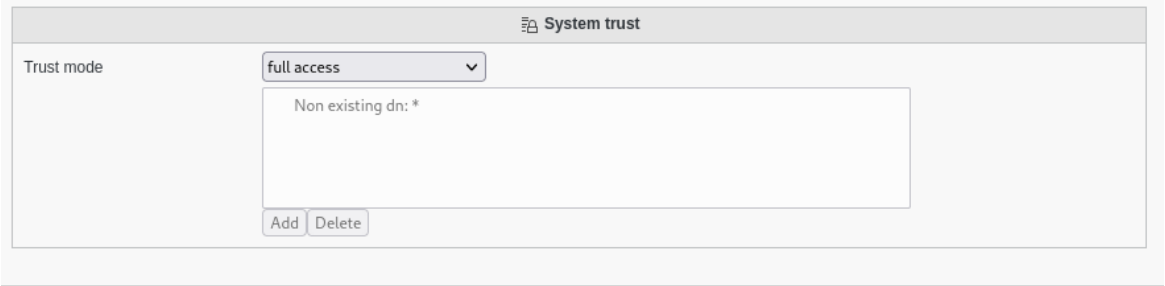


Figura 4.6 Atributo System Trust, del Plugin POSIX de FusionDirectory

En la Figura 4.4 se pueden ver los usuarios añadidos al árbol LDAP, mientras que en la Figura 4.5 se pueden observar los sistemas añadidos, correspondientes a los equipos de la red. Por otra parte, en la Figura 4.6 se puede observar uno de los atributos POSIX: System Trust. Mediante este atributo se pueden gestionar los equipos a los que tiene acceso un usuario. Cuando se otorga acceso a un equipo con este método, se añade el atributo *host* al objeto inetOrgPerson. Este atributo estándar contendrá el nombre de todos los equipos a los que puede acceder el usuario, o "*****" si tiene acceso a todos los equipos. Esta característica puede ser utilizada para gestionar el acceso a los diferentes equipos de la red. De este modo, una persona solo tendrá acceso al equipo que le ha sido asignado, mientras que el administrador tendrá pleno acceso. Otra opción posible, que puede resultar útil circunstancialmente es deshabilitar el acceso del usuario a cualquier host. Esta opción eliminará el atributo mencionado, lo que supondrá que el usuario no pueda acceder a ningún equipo. Gracias a esta estructura del DIT, una aplicación de gestión de acceso podrá

permitir o restringir el acceso de estos usuarios al equipo de forma sencilla. La aplicación que se utilizará para gestionar el acceso a los equipos será SSSD (System Security Services Daemon), herramienta open source, compatible con LDAP, de uso extendido y con amplia documentación y comunidad de usuarios [25]. Esta herramienta, a través de directivas en su archivo de configuración se puede personalizar para realizar autenticación a través de LDAP y utilizar el atributo host para comprobar el acceso al equipo.

```
[sssd]
config_file_version = 2
domains = ldap

[domain/ldap]
id_provider = ldap
auth_provider = ldap
chpass_provider = ldap
access_provider = ldap
ldap_uri = ldap://raton.redldap.es
ldap_search_base = dc=redldap,dc=es
ldap_access_order = host
```

Figura 4.7 Archivo de configuración SSSD

En la Figura 4.7 se puede ver la configuración escogida para SSSD. En este destacan los atributos access_provider y ldap_access_order. Con el primero se indicará al demonio que se consultará un servicio LDAP para comprobar el acceso de los usuarios. Con el segundo se indicará que la comprobación de acceso se realizará con el atributo host del usuario. Por tanto, configurando adecuadamente el DIT, el nombre de host en cada equipo e instalando el servicio SSSD se consigue que solo unos usuarios determinados puedan acceder a un host concreto. De este modo, se cumple el objetivo planteado inicialmente, ya que un usuario tendrá acceso solo a su host, imposibilitando que otros usuarios, que no sean el administrador, acceder al mismo.

Para sintetizar, en la Tabla 1 se detallan todas las herramientas utilizadas para construir la red, con sus correspondientes versiones.

Tabla 1 Software Instalado

Software	**Servicio ofrecido**	**Equipos**	**Versión**
Debian	Sistema Operativo	LDAPServer	11
Debian	Sistema Operativo	Hosts clientes y Router	12
ISC DHCP Server	DHCP	Router	4.4.3-P1
ISC DHCP Client	DHCP	Hosts clientes y LDAPServer	4.4.1
BIND9	DNS	Router	9.18.19-1~deb12u1-Debian
SSSD	Autenticación	Hosts clientes y LDAPServer	2.4.1
OpenLDAP: slapd	Servidor LDAP	LDAPServer	2.4.57+dfsg-3+deb11u1
FusionDirectory	Aplicación de gestión LDAP	LDAPServer	1.4

Una vez configurada la red y el acceso de los usuarios a esta surge el problema central de esta investigación: detectar usuarios que intentan acceder a equipos a los que no poseen acceso. En el capítulo siguiente se detallarán los pasos seguidos para desarrollar sobre la red una herramienta capaz de detectar estas acciones y otros comportamientos potencialmente maliciosos.

5. Desarrollo de herramienta de control de acceso a la red

Una vez creada la red se definirá el proceso de desarrollo de la herramienta que permitirá a los administradores detectar comportamientos sospechosos en la misma, por ejemplo, el acceso de un usuario a un equipo ajeno. Para ello, ni el protocolo LDAP ni ninguna de las aplicaciones instaladas ofrecen una solución directa, por lo que se hará uso de múltiples elementos de los recursos descritos en el capítulo anterior para resolver esta problemática.

En el desarrollo de la solución propuesta, se pueden destacar tres etapas bien diferenciadas. La primera etapa consiste en la creación del entorno de desarrollo, la cual ya ha sido tratada en la sección 4. En la siguiente etapa, tratada en la sección 5.1, se buscará poder registrar los eventos que ocurren en la red. Esta información será procesada mediante un script, dando lugar a un archivo que pueda ser tratado con facilidad por la aplicación. En último lugar, en la sección 5.2 se desarrollará una interfaz gráfica y una REST API que consultarán este archivo para mostrar y filtrar la información contenida en él.

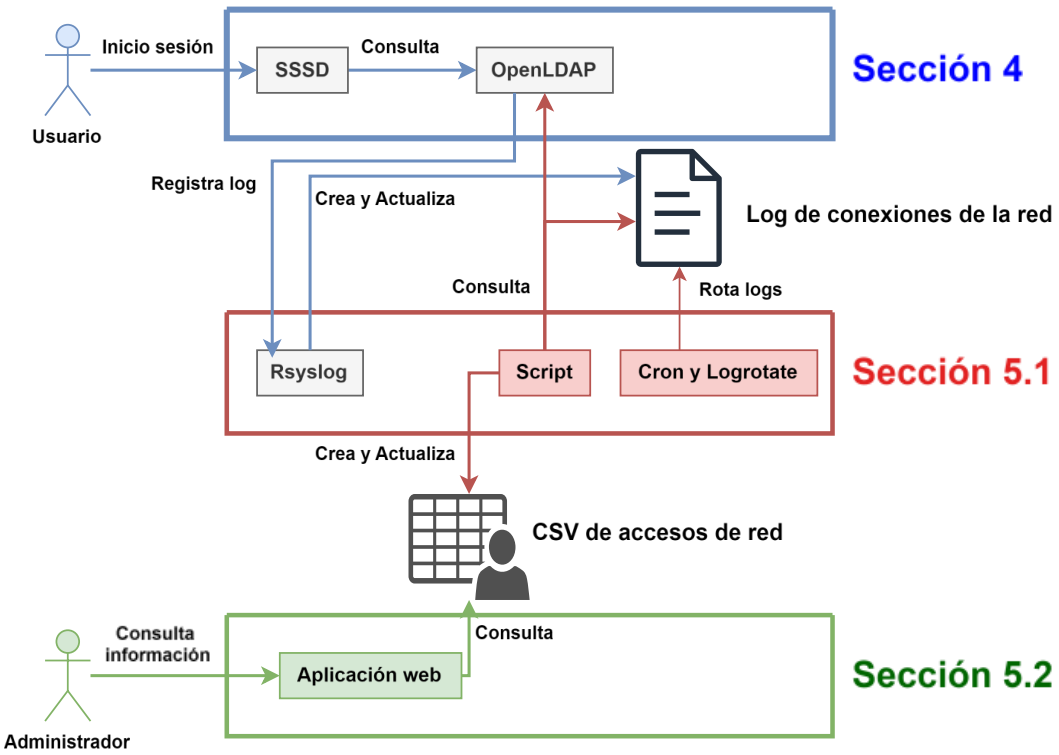


Figura 5.1 Diagrama de componentes aplicación

En la Figura 5.1 se puede observar un diagrama de componentes que muestra los diferentes elementos que componen la solución final. También, se indica la sección del proyecto en la que son implementados y explicados en profundidad y las interacciones existentes entre los distintos elementos. Por tanto, gracias al esquema mostrado se puede tener una visión global acerca de la solución elaborada.

5.1 Auditoría

Para comenzar, se necesitan registrar las conexiones realizadas al servidor LDAP mediante un sistema de logs o alguna funcionalidad similar. Esto es ofrecido, entre otras muchas opciones, por el plugin Audit de FusionDirectory [6]. Se valora positivamente esta opción por su facilidad de uso y de instalación. Pese a ello, este sistema no ofrece la funcionalidad deseada, pues solo registra las conexiones y la actividad realizada en la propia aplicación de FusionDirectory, no con todo el servidor LDAP. Por tanto, esta opción puede considerarse como complementaria a la solución final, pero no como medio para resolver el problema planteado. Otra opción que podría ser clave para conseguir implementar esta solución es Accesslog, un overlay de OpenLDAP [1], [26]. Esta adición al servidor permite registrar y almacenar en el DIT, junto con resto de información del mismo, todos los accesos realizados sobre una base de datos concreta del servidor LDAP. En contraposición con el plugin de FusionDirectory, este overlay posee un proceso de instalación algo más complejo, pues se debe realizar mediante los comandos ofrecidos por OpenLDAP. Una vez instalado, se puede configurar para que se adapte a los requisitos del usuario, por lo que se puede concluir que existe flexibilidad en su uso. Por ejemplo, se puede configurar para registrar solo las operaciones deseadas o para eliminar los logs cada cierto tiempo.

 # 20240205190509.000002Z, accesslog

dn: reqStart=20240205190509.000002Z,cn=accesslog

objectClass: auditBind

reqStart: 20240205190509.000002Z

reqEnd: 20240205190509.000003Z

reqType: bind

reqSession: 1006

reqAuthzID:

reqControls: {0}{1.3.6.1.4.1.42.2.27.8.5.1}

reqDN: uid=userone,ou=people,dc=redldap,dc=es

reqResult: 0

reqVersion: 3

reqMethod: SIMPLE

Figura 5.2 Log de AuditLog overlay

En la Figura 5.2 se puede observar el log del establecimiento de una conexión por parte de userone. Ofrece gran cantidad de información útil, como la fecha o el DN del usuario que realiza la conexión, pero como se observa no ofrece información alguna acerca de la máquina desde la que se ha realizado la conexión. Por tanto, esta alternativa, pese a su flexibilidad y gran utilidad, no permite lograr el objetivo planteado. Otra opción potencialmente válida es hacer uso de los propios logs del servidor LDAP. Estos pueden ser activados de forma muy sencilla, simplemente variando el atributo *olcLogLevel* del servidor, mediante un comando. El propio sistema de logs que ofrece OpenLDAP puede almacenar una gran variedad de operaciones realizadas en el servidor, dependiendo del valor asignado al atributo mencionado [1]. A la hora de asignarle valor a este atributo, se debe tener en cuenta que los distintos niveles de log registran información diferente. Cada nivel es completamente independiente del resto, por lo que establecer un nivel superior no incluirá toda la información de los niveles anteriores. También es importante resaltar que se pueden establecer varios niveles de log al mismo tiempo. Por tanto, se tratará de establecer el valor mínimo para cumplir con los objetivos planteados, evitando así guardar información adicional que no sirva para desarrollar la solución requerida. En concreto, se optará por asignarle el valor 256, correspondiente al nivel *stats*. Con este nivel, se almacenarán conexiones, operaciones y resultados en el servidor.

Feb 21 18:47:42 LDAPServer slapd[1815]: ***conn=1001*** *op=5 SRCH base="****dc=redldap,dc=es****" scope=2 deref=0*

filter="(&(uid=userone)(objectClass=posixAccount)(&(uidNumber=*)(!(uidNumber=0))))"

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1001 op=5 SRCH attr=objectClass uid userPassword uidNumber gidNumber gecos homeDirectory loginShell krbPrincipalName cn modifyTimestamp modifyTimestamp shadowLastChange shadowMin shadowMax shadowWarning shadowInactive shadowExpire shadowFlag krbLastPwdChange krbPasswordExpiration pwdAttribute authorizedService accountExpires userAccountControl nsAccountLock host rhost loginDisabled loginExpirationTime loginAllowedTimeMap sshPublicKey userCertificate;binary mail

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1001 op=5 SEARCH RESULT tag=101 err=0 nentries=1 text=

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1001 op=6 SRCH base="dc=redldap,dc=es" scope=2 deref=0 filter="(&(memberUid=userone)(objectClass=posixGroup)(cn=)(&(gidNumber=*)(!(gidNumber=0))))"*

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1001 op=6 SRCH attr=objectClass cn userPassword gidNumber modifyTimestamp modifyTimestamp

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1001 op=6 SEARCH RESULT tag=101 err=0 nentries=0 text=

Figura 5.3 Log de operación de búsqueda LDAP

Feb 21 18:47:42 LDAPServer slapd[1815]: ***conn=1003*** *fd=18 ACCEPT from* ***IP=192.168.3.10:33890***

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1003 op=0 EXT oid=1.3.6.1.4.1.1466.20037

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1003 op=0 STARTTLS

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1003 op=0 RESULT oid= err=0 text=

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1003 fd=18 TLS established tls_ssf=256 ssf=256

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1003 op=1 SRCH base="" scope=0 deref=0 filter="(objectClass=)"*

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1003 op=1 SRCH attr= altServer namingContexts supportedControl supportedExtension supportedFeatures supportedLDAPVersion supportedSASLMechanisms domainControllerFunctionality defaultNamingContext lastUSN highestCommittedUSN*

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1003 op=1 SEARCH RESULT tag=101 err=0 ***nentries=1*** *text=*

*Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1003 op=2 BIND dn="****uid=userone,ou=people,dc=redldap,dc=es****" method=128*

Feb 21 18:47:42 LDAPServer slapd[1815]: slap_global_control: unrecognized control: 1.3.6.1.4.1.42.2.27.8.5.1

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1003 op=2 RESULT tag=97 ***err=49*** *text=*

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1003 op=3 UNBIND

Feb 21 18:47:42 LDAPServer slapd[1815]: conn=1003 fd=18 closed

Figura 5.4 Log de bind de búsqueda LDAP

En los logs representados en la Figura 5.3 y 5.4 se encuentra la clave para desarrollar la solución deseada. Para comenzar, al realizar una operación de inicio de sesión, un bind, la máquina cliente completa dos acciones principales. La primera, de búsqueda, viene representada por la Figura 5.3. Estas operaciones de búsqueda, previas al bind, se realizan de forma anónima desde la máquina cliente. Primero la máquina cliente establece una conexión LDAP con el servidor. La identidad de esta conexión está determinada por el parámetro *conn*, siendo su valor en este caso de 1001. Sucesivas operaciones realizadas por el cliente desde esta máquina utilizarán esta conexión. Por ejemplo, si realiza otros inicios de sesión tras el primero, las búsquedas previas al bind serán todas realizadas con esta conexión cuyo valor de *conn* será 1001. En estos logs se puede obtener además otra información relevante, la cual está destacada en negrita en la figura para facilitar su visualización. Destaca, además de la id de conexión y la región del DIT que funciona como base de la búsqueda, la entrada buscada en la operación. En el caso de una operación de inicio de sesión, la entrada buscada será la del usuario que se quiere autenticar, en este caso, userone. En la Figura 5.4 se recogen los logs de una operación de bind, que,

junto con la búsqueda previa, completaría el inicio de sesión. Se puede observar que tiene una id de conexión diferente a la de la búsqueda previa. Esto ocurrirá en todos los casos, estableciéndose siempre una nueva conexión LDAP para hacer la operación de bind. También, se recoge la IP desde la cual se ha realizado la operación, lo cual es clave para desarrollar el objetivo del proyecto. Se debe aclarar que, en el caso de la conexión establecida para las búsquedas, también se indica la IP cuando se establece la conexión, pero no cada vez que se realiza una búsqueda. Además, una vez más, se puede observar el DN del usuario que se está autenticando, Por último, también se observa el código de error de la operación, lo que puede indicar ciertos errores de conexión u otra clase de fallos. En este caso, el error 49 indica que el usuario introdujo una contraseña errónea [27]. Con toda esta información, se puede desarrollar un script para procesar estos logs y extraer la información más relevante de ellos, para presentarla de forma clara y organizada.

Antes de comenzar a elaborar el script de procesamiento de logs, se deben redirigir los mismos a un archivo de texto, para que sea más sencilla su manipulación, pues por defecto OpenLDAP almacena sus logs en la aplicación Rsyslog [28] de Linux. Desde la configuración de dicho programa se deben redirigir los logs de slapd al archivo de texto deseado, que se llamará ldap.log. Una vez completada esta acción todos los logs de OpenLDAP serán almacenados en el archivo, por lo que un programa podrá hacer uso de esta información fácilmente. A continuación, se desarrollará el script encargado de procesar este archivo para poder almacenar toda esta información de forma ordenada y clara. El formato de almacenamiento de esta información puede variar según las necesidades y preferencias del usuario. En este proyecto se utilizará un archivo en formato .csv para ello, por su gran estandarización, facilidad de exportación y su compatibilidad con otros programas.

El script será construido con el lenguaje de programación Python por su facilidad de uso y su gran cantidad de bibliotecas, utilizadas para construir el CSV y la posterior aplicación. Para construir las diferentes conexiones e intentos de inicio de sesión, se utilizará como base la estructura de logs mencionada anteriormente en las Figuras 5.3 y 5.4. Como se indicó anteriormente para establecer una conexión como tal, se debe producir una búsqueda previa y un

bind a continuación. De este modo, el script podrá diferenciar otro tipo de conexiones no orientadas a inicio de sesión. Por ejemplo, una simple búsqueda realizada con ldapsearch no será detectada como inicio de sesión, pues no contará con el posterior bind. Con estas premisas, el script analizará el log línea por línea y cuando se cumplan las condiciones mencionadas se habrá detectado un inicio de sesión. El código completo del script de procesamiento de logs se encuentra en el repositorio Github indicado en el Anexo A.

conn	start_time	username	statusUser	ip_address	admin	host	has_access	codError
1003	2024-05-12 19:57:05	userone	LDAP	192.168.3.10	False	LDAPServer.redldap.es.	False	0
1004	2024-05-12 19:57:21	sergioat	LDAP	192.168.3.10	True	LDAPServer.redldap.es.	True	0

Figura 5.5 Ejemplo CSV con dos accesos

Para cada conexión se almacenará información potencialmente útil para un administrador: id de conexión, fecha de conexión, IP, nombre de host, usuario, tipo de usuario, acceso al equipo, código de error y estatus de administrador. En la Figura 5.5 se pueden observar dos columnas del archivo, generadas tras detectar dos accesos.

-La id de conexión, la fecha de conexión, la IP, el usuario y el código de error se pueden obtener directamente de los logs, ya que se indican explícitamente en el formato de logs de OpenLDAP, explicado anteriormente.

-El atributo tipo de usuario indicará si el usuario que intenta acceder al equipo pertenece a los usuarios almacenados en LDAP o no. Esto es debido a que, en caso de existir usuarios locales en las máquinas, realizarán también una búsqueda LDAP, sin el bind posterior que realizaría un usuario registrado en el servidor. Además, en la búsqueda, el atributo nentries tendrá un valor de 0, ya que no se han encontrado coincidencias en el servidor. La misma situación ocurre si se intenta iniciar sesión con un usuario inexistente. De este modo, se pueden distinguir dos clases de usuarios: los usuarios registrados en LDAP y los ajenos a LDAP.

-El atributo estatus de administrador indicará si el usuario que realizó el intento de inicio de sesión pertenece al grupo de administradores. Gracias a FusionDirectory, se pueden gestionar de forma sencilla los permisos de administrador, asignando las ACLs a usuarios mediante ACL assignments. Estas asignaciones funcionan como grupos, otorgándole los permisos definidos para

el grupo a los usuarios pertenecientes al mismo. De este modo, el script mediante una búsqueda LDAP se comprueba si el usuario pertenece al grupo “admin”, que es el grupo al que pertenecen los administradores. Si el usuario que realizó la operación se encuentra en el grupo será administrador, siendo no administrador en caso contrario.

-El atributo acceso al equipo comprobará si el usuario tenía acceso efectivo al equipo en el cual ha intentado iniciar sesión. Esto se comprobará en el script realizando una búsqueda LDAP al servidor para localizar el atributo *host* del usuario. Si este atributo existe y coincide con el nombre de host desde el cual se ha establecido la conexión se concluirá que el usuario tenía permisos de acceso. También se considerará que el usuario tiene acceso si el atributo host tiene el valor ‘*’, correspondiente a pleno acceso a todos los equipos. La obtención del nombre del host se realizará a través de una consulta DNS inversa, mediante la IP obtenida previamente. En caso de que el atributo host no exista en el usuario, lo que equivale a no tener acceso a ningún equipo o no coincida su valor con el del nombre de host del equipo que realizó la conexión, se habrá detectado un posible acceso fraudulento del usuario a un equipo para el cual no tiene permisos de acceso. Con esta funcionalidad se habrá logrado resolver el principal problema planteado en este proyecto: la detección de usuarios que intentan acceder a equipos para los cuales no disponen de permisos.

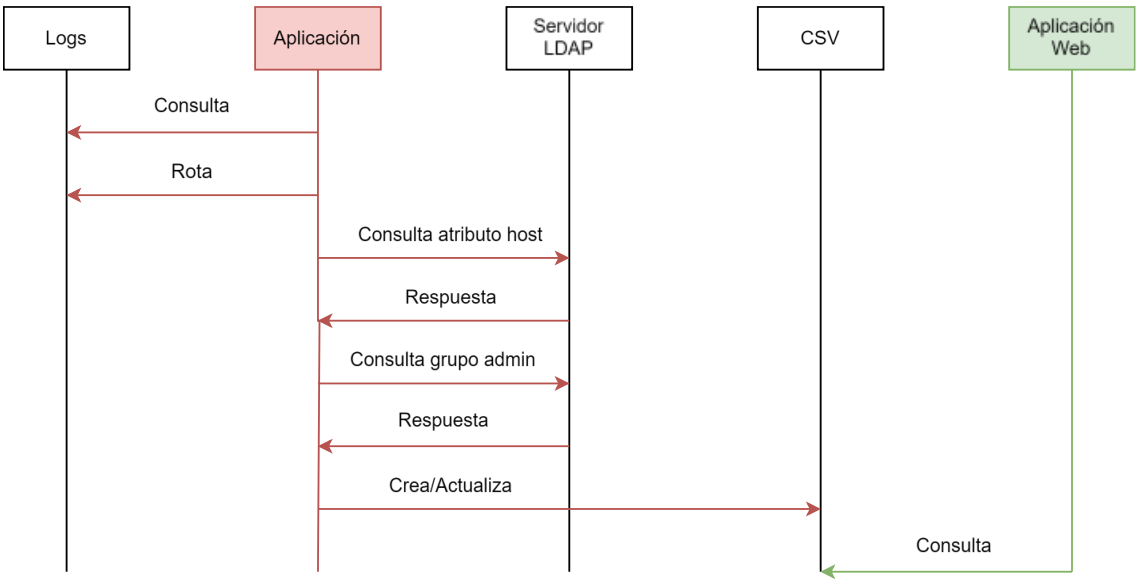


Figura 5.6 Diagrama de secuencia aplicación

El comportamiento de la aplicación, es decir, del script y del sistema creado para la rotación de los logs se muestra en la Figura 5.6. El script se encargará de revisar el archivo de texto que posee los logs, buscando el patrón búsqueda-bind, que realiza un usuario al intentar iniciar sesión. Por otra parte, los logs rotarán periódicamente para evitar saturación en su almacenamiento. Del archivo mencionado se extrae toda la información potencialmente relevante para un administrador, además de recopilar otros datos relevantes como el acceso o si el usuario es administrador, mediante consultas al servidor. Finalmente, todos estos intentos de acceso son guardados en el archivo CSV, para facilitar su consulta. Con esta herramienta un administrador sería capaz de detectar accesos no autorizados, conexiones sospechosas y actividad potencialmente maliciosa, simplemente consultando este archivo. Para facilitar la consulta de esta información, las siguientes acciones estarán encaminadas a ofrecer una interfaz gráfica cómoda e intuitiva para poder obtener estos datos fácilmente.

5.2 REST API e interfaz gráfica

Primero, para evitar que el archivo de logs aumente su tamaño de forma indefinida, se hará uso de logrotate y cron[29-30], con el objetivo de que el archivo de log rote semanalmente y para ejecutar el script de procesamiento de logs periódicamente. Además, se programará un reinicio del servidor LDAP junto con la rotación de los logs, para cerrar todas las conexiones que pudieran haber quedado abiertas y así evitar la complicación que supondría tener varias conexiones a medias en varios archivos de log diferentes. A continuación, se creará una REST API, que servirá para poder consultar el archivo de forma sencilla. Para desarrollar esta API se hará uso de Flask, un framework open-source de Python que permite la creación sencilla y rápida de aplicaciones web [31].

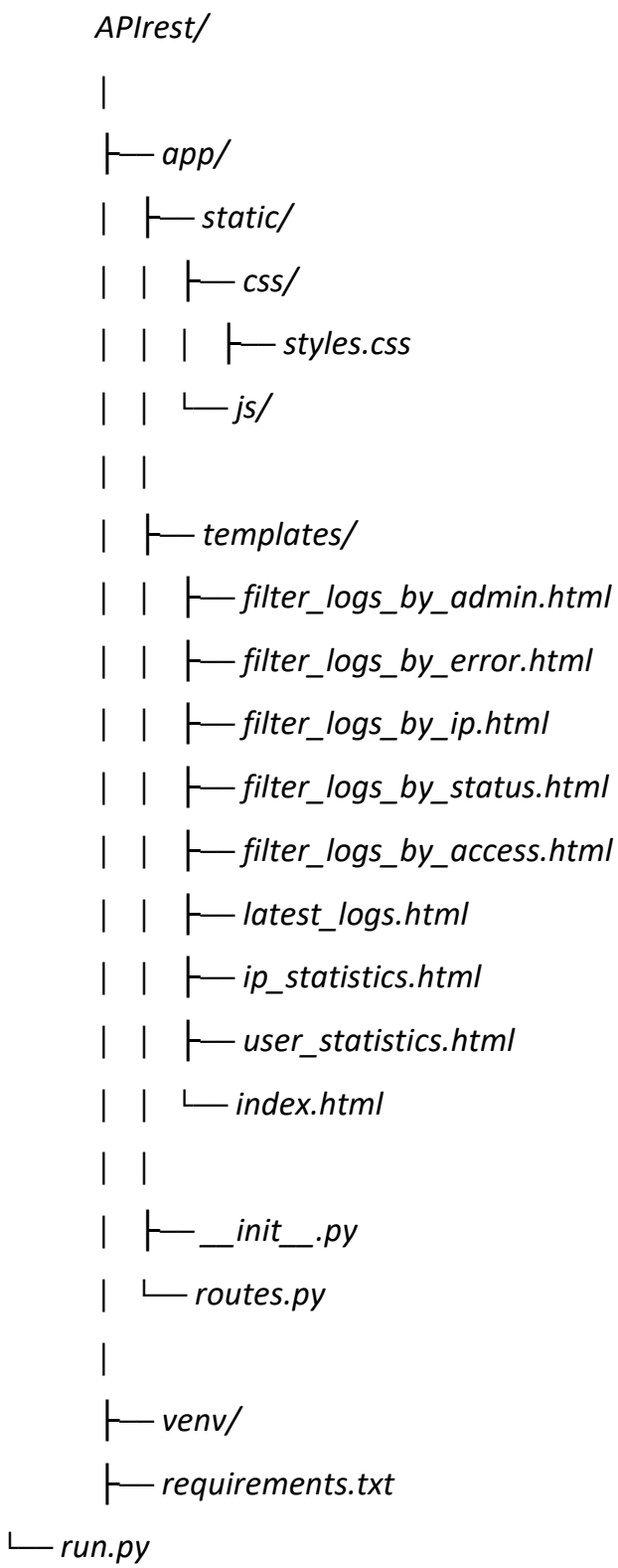


Figura 5.7 Estructura de carpetas aplicación

En la Figura 5.7 se puede ver la estructura de carpetas utilizada en la aplicación. Se utilizará un entorno virtual para evitar incompatibilidades en la instalación de los distintos programas necesarios. Estas instalaciones estarán indicadas en el

archivo requirements.txt. Run.py contendrá el código, escrito en Python, necesario para iniciar el servicio y routes.py contendrá los *endpoints* del API REST, es decir, las distintas peticiones que el usuario podrá realizar a la aplicación. Estas peticiones serán procesadas y tras hacer una consulta al archivo CSV mostrarán por pantalla de forma ordenada la información solicitada. Una vez construido el REST API, el usuario, además de realizar peticiones a través de peticiones manuales al servidor, puede hacer uso de la interfaz gráfica web, para una mayor comodidad. Estas páginas se construyen a partir de los archivos html de la carpeta templates, y su estilo estará determinado por el código css del archivo styles.css. En el Anexo A se incluye un enlace al repositorio Github que contiene todos los archivos mencionados en esta sección, además del script de la sección anterior.

Figura 5.8 Menú aplicación web

Figura 5.9 Filtro de logs por acceso

Una vez que se accede a la aplicación web desde el buscador, se mostrará un menú como el mostrado en la Figura 5.8. Desde los distintos botones se podrá acceder a las funcionalidades de filtrado de información que ofrece la REST API. Al acceder a cada sección, el usuario será redirigido a otra página, en la que se indicarán los inicios de sesión ocurridos en un formato tabular. De este modo el usuario podrá filtrar los logs por los distintos atributos que posee un inicio de sesión como IP, nombre de host o código de error. Además, también podrá recopilar estadísticas de uso de usuario e IP. Esto le permitirá observar el

número de conexiones que ha realizado un usuario o se han realizado desde una IP concreta en un periodo de tiempo determinado. En la Figura 5.9 se muestra un ejemplo de filtro por acceso, en el que se buscan los inicios de sesión en los que no hay autorización de acceso (el checkbox del filtro aparece como “true” ya que al filtrar se actualiza la página). Gracias a este filtro el administrador puede ver que userone ha intentado iniciar sesión en un equipo para el que no cuenta con acceso. De esta forma, el administrador podrá actuar en consecuencia del comportamiento potencialmente malicioso observado.

En conclusión, mediante la aplicación web, construida gracias a la información recopilada en el archivo CSV, un administrador podrá observar los accesos de los diferentes usuarios de la red a los equipos de la misma. Mediante los filtros implementados podrá clasificar de forma rápida y cómoda la información requerida, como por ejemplo filtrar por el nombre de usuario o solo mostrar los accesos en una fecha determinada. Así queda cumplido el objetivo fundamental planteado en el proyecto: utilizar el protocolo LDAP para detectar accesos indebidos a los equipos de la red.

Conclusiones

LDAP es un protocolo que puede resultar de gran utilidad para centralizar la información de usuarios en una red, gracias a su modo de estructurar la información almacenada y a su gran eficiencia en consultas. De esta forma, se puede gestionar el acceso de los diferentes usuarios a los equipos de forma cómoda y sencilla. Sin embargo, LDAP tiene importantes problemas en materia de seguridad informática que deben ser tenidos en cuenta para su adecuada utilización. Además de ser vulnerable a ataques comunes como interceptación de contraseñas o denegación de servicio, posee varios ataques específicos, como LDAP injection o LDAP Blind injection. En adición, si se quiere utilizar LDAP para centralizar la información de usuarios en la red con el objetivo de proveer servicios de autenticación y acceso, surgen otra serie de problemas que el protocolo no solventa. Entre estos problemas destaca la ausencia de métodos para poder detectar comportamientos potencialmente maliciosos, en especial, los intentos de acceso de usuarios a equipos para los cuales no poseen acceso. Otros problemas relacionados para los que tampoco existe una solución clara por parte de las implementaciones de LDAP son la detección de conexiones de IPs sospechosas o un número excesivamente alto de intentos de acceso con contraseñas incorrectas. En definitiva, LDAP no ofrece un modo sencillo de poder monitorear los diferentes eventos que ocurren en la red, en especial eventos que puedan implicar potenciales ataques.

Como respuesta, se ha desarrollado una herramienta que se encarga de paliar estos problemas. Esta ofrece a un administrador información acerca de todos los accesos e intentos de acceso ocurridos en la red. Esta información se presenta de forma gráfica y ordenada, dando la posibilidad al administrador de filtrar la misma según varios criterios a su elección, como la fecha de los accesos o mostrar las conexiones en las que un usuario accedió a un equipo sin permiso. De esta forma el administrador puede detectar comportamientos sospechosos y tomar medidas en consecuencia para evitar posibles ataques. En consecuencia, las mejoras en seguridad que ofrece la herramienta desarrollada pueden dar lugar a redes más seguras, en las que un administrador podrá hacer frente a ataques de forma más cómoda. Por ejemplo, su uso en el ámbito empresarial o universitario, dos ámbitos en los que se podría hacer uso de LDAP para

centralizar cuentas de usuario, podría ahorrar muchos costes debidos a ataques informáticos y problemas de seguridad, ya que un administrador podría detener estos ataques a tiempo gracias a contar con información de calidad sobre los accesos que ocurren en la red. Debido a lo importante que resulta la prevención en el diseño y gestión de sistemas seguros, es esencial contar con herramientas como esta, que ofrecen un modo centralizado, sencillo y visual de obtener información relevante para un administrador.

Trabajo futuro

La herramienta propuesta ofrece una solución ante las diferentes carencias en materia de seguridad mencionadas que posee el protocolo LDAP. No obstante, partiendo de la aplicación desarrollada como punto de partida, se pueden añadir futuras actualizaciones y nuevas características con el objetivo de ampliar sus funcionalidades. Para comenzar, un modo de aumentar la seguridad ofrecida sería complementarla con otras herramientas de seguridad. Por ejemplo, el overlay Password Policies [1] de OpenLDAP ofrece al administrador la capacidad de introducir políticas de contraseñas, como longitud y complejidad mínima o tiempos de espera y bloqueos de cuenta en caso de fallos reiterados. Otra funcionalidad que se complementa a la perfección con este sistema es la generación de perfiles estadísticos de usuario por parte de una aplicación externa. Estos perfiles consisten en la recopilación de datos acerca de las horas más frecuentes en las que un usuario trabaja en su equipo. Cuando se detecta una conexión fuera de esas horas establecidas, lo que podría indicar comportamiento sospechoso, se detectaría en la herramienta como conexión a deshora. Adicionalmente, al detectar esta clase de comportamientos sospechosos, se podría activar una alerta que avise al administrador en tiempo real, por ejemplo, enviándole un correo o activando una notificación del sistema. Esto se podría lograr modificando el script para ejecutar estas alertas al procesar esta clase de conexiones sospechosas, como las ya mencionadas conexiones a deshora, conexiones a equipos por parte de usuarios que no tienen permiso de acceso o conexiones desde IPs externas a la red. Este sistema de alertas podría incluir también el bloqueo de la IP o la cuenta que provocó la alerta, ante este tipo de comportamientos. Con estas adiciones se podrá contar con una

herramienta que añada aún más seguridad a sistemas de acceso que utilicen LDAP como base. Finalmente, la funcionalidad otorgada por la herramienta creada en este proyecto debería ser incluida en un futuro en las diferentes implementaciones o aplicaciones del protocolo LDAP, con el objetivo de mitigar sus carencias en seguridad, en este caso, en su uso para centralizar la información de usuarios.

Conclusions

LDAP is a protocol that can be very useful for centralizing user information in a network, thanks to its way of structuring the stored information and its great efficiency in queries. In this way, it is possible to manage the access of different users to computers in a convenient and simple way. However, LDAP has important problems in terms of computer security that must be taken into account for its proper use. In addition to being vulnerable to common attacks such as password interception or denial of service, it has several specific attacks, such as LDAP injection or LDAP Blind injection. In addition, if you want to use LDAP to centralize user information on the network in order to provide authentication and access services, another series of problems arise that the protocol does not solve. Among these problems is the absence of methods for detecting potentially malicious behavior, in particular, attempts by users to access equipment to which they do not have access. Other related problems for which there is also no clear solution from LDAP implementations are the detection of connections from suspicious IPs or an excessively high number of access attempts with incorrect passwords. In short, LDAP does not offer an easy way to monitor the different events occurring on the network, especially events that may involve potential attacks.

In response, a tool has been developed to alleviate these problems. It provides an administrator with information about all accesses and access attempts that have occurred on the network. This information is presented in a graphical and ordered form, giving the administrator the possibility to filter it according to various criteria of his choice, such as the date of the accesses or the connections that are not allowed. In this way, the administrator can detect suspicious behavior and take action accordingly to prevent possible attacks. As a result, the security improvements offered by the developed tool can lead to more secure networks, in which an administrator will be able to deal with attacks more comfortably. For example, its use in the corporate or university environment, two areas where LDAP could be used to centralize user accounts, could save a lot of costs due to cyber attacks and security problems, since an administrator could stop these attacks in time thanks to having quality information about the accesses occurring

in the network. Because of the importance of prevention in the design and management of secure systems, it is essential to have tools like this, which offer a centralized, simple and visual way to obtain relevant information for an administrator.

Future work

The proposed tool offers a solution to the various security shortcomings of the LDAP protocol. However, starting from the developed application as a base point, future updates and new features can be added in order to extend its functionalities. To begin with, one way to increase the security offered would be to complement it with other security tools. For example, OpenLDAP's Password Policies overlay [1] offers the administrator the ability to enter password policies, such as minimum length and complexity or timeouts and account lockouts in case of repeated failures. Another functionality that perfectly complements this system is user profiling by an external application. This profiling consist of collecting data about the most frequent times a user works on his or her computer. When a connection is detected outside these established hours, which could indicate suspicious behavior, it would be detected in the tool as an after-hours connection. Additionally, when detecting this kind of suspicious behavior, an alert could be triggered to warn the administrator in real time, for example, by sending an email or activating a system notification. This could be achieved by modifying the script to execute these alerts when processing this kind of suspicious connections, such as the aforementioned after-hours connections, connections to computers by users who do not have access permission or connections from IPs outside the network. This alert system could also include the blocking of the IP or account that triggered the alert. With these additions, it will be possible to have a tool that adds even more security to access systems that use LDAP as a base. Finally, the functionality provided by the tool created in this project should be included in the future in the different implementations or applications of the LDAP protocol, in order to mitigate its security shortcomings, in this case, in its use to centralize user information.

Índice de Acrónimos

LDAP: **L**ightweight **D**irectory **A**ccess **P**rotocol. Protocolo estándar para el acceso a información de directorios.

TCP/IP: **T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol. Modelo de conexiones de red utilizado en Internet.

DAP: **D**irectory **A**ccess **P**rotocol. Protocolo predecesor de LDAP.

TLS: **T**ransport **L**ayer **S**ecurity. Protocolo que implementa conexiones seguras en Internet.

IP: **I**nternet **P**rotocol. Protocolo de la capa de red TCP/IP.

API REST: **A**pplication **P**rogramming **I**nterface **RE**presentational **S**tate **T**ransfer. Interfaz capaz de interactuar con un servicio que implementa la arquitectura REST.

OSI: **O**pen **S**ystems **I**nterconnection. Modelo estándar de conexiones de red.

SQL: **S**tructured **Q**uery **L**anguage. Lenguaje utilizado en la gestión de bases de datos relacionales.

OID: **O**bject **ID**. Identificacor de un objeto LDAP a nivel global.

RFC: **R**equest **F**or **C**ommons. Documentación técnica de la organización IETF.

IETF: **I**nternet **E**ngineering **T**ask **F**orce: Organización dedicada a la estandarización y desarrollo de Internet.

DIT: **D**irectory **I**nformation **T**ree. Estructura de árbol en la que se organiza la información en LDAP.

DN: **D**istinguished **N**ame: Nombre que identifica de forma única a un objeto en la jerarquía LDAP, formado por la unión de RDNs.

RDN: **R**elative **D**istinguished **N**ame: Nombre característico en una entrada LDAP, formado por un atributo único en la jerarquía o una combinación única de varios atributos.

DNS: **D**omain **N**ame **S**ystem: Protocolo de resolución de nombres de dominio.

LDIF: **L**DAP **D**ata **I**nterchange **F**iles: Formato de representación de datos de la información de un directorio.

ACL: **A**ccess **C**ontrol **L**ist. Listas de control de acceso, utilizadas para gestionar permisos de usuarios.

DOS: **D**enial **O**f **S**ervice. Ataque de denegación de servicio.

DDOS: **D**istributed **D**enial **O**f **S**ervice. Ataque de denegación de servicio distribuido.

POSIX: **P**ortable **O**perating **S**ystem **I**nterface for UNI**X**. Estándares para compatibilidad y portabilidad en sistemas UNIX [24].

SSSD: **S**ystem **S**ecurity **S**ervices **D**aemon. Herramienta de gestión de acceso de usuarios a un sistema.

Bibliografía

[1] «OpenLDAP Software 2.6 Administrator's Guide». Accedido: 11 de abril de 2024. [En línea]. Disponible en: https://www.openldap.org/doc/admin26/

[2] «X.500 : Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services». Accedido: 6 de mayo de 2024. [En línea]. Disponible en: https://www.itu.int/rec/T-REC-X.500-201910-I/en

[3] «Open Source Guide - LDAP for Rocket Scientists - Contents». Accedido: 11 de abril de 2024. [En línea]. Disponible en: https://www.zytrax.com/books/ldap/

[4] R. Harrison, «Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms», Internet Engineering Task Force, Request for Comments RFC 4513, jun. 2006. doi: 10.17487/RFC4513.

[5] «Softerra LDAP Administrator & Browser: Directory Management Tool for Windows». Accedido: 16 de abril de 2024. [En línea]. Disponible en: https://www.ldapadministrator.com/

[6] «Introduction to FusionDirectory — FusionDirectory User Manual 1.4 documentation». Accedido: 16 de abril de 2024. [En línea]. Disponible en: https://fusiondirectory-user-manual.readthedocs.io/en/1.4/index.html

[7] L. Li y W. Chou, «Design and Describe REST API without Violating REST: A Petri Net Based Approach», en 2011 IEEE International Conference on Web Services, jul. 2011, pp. 508-515. doi: 10.1109/ICWS.2011.54.

[8] J. Sermersheim, «Lightweight Directory Access Protocol (LDAP): The Protocol», Internet Engineering Task Force, Request for Comments RFC 4511, jun. 2006. doi: 10.17487/RFC4511.

[9] «About RFCs», IETF. Accedido: 17 de abril de 2024. [En línea]. Disponible en: https://www.ietf.org/process/rfcs/

[10] B. Varanasi y A. Sacco, «Introduction to LDAP», en Practical Spring LDAP: Using Enterprise Java-Based LDAP in Spring Data and Spring Framework 6, B.

Varanasi y A. Sacco, Eds., Berkeley, CA: Apress, 2023, pp. 1-18. doi: 10.1007/979-8-8688-0002-3_1.

[11] M. C. Smith, «Definition of the inetOrgPerson LDAP Object Class», Internet Engineering Task Force, Request for Comments RFC 2798, abr. 2000. doi: 10.17487/RFC2798.

[12] R. Huber, S. R. Sataluri, A. Grimstad, M. Wahl, y S. Kille, «Using Domains in LDAP/X.500 Distinguished Names», Internet Engineering Task Force, Request for Comments RFC 2247, ene. 1998. doi: 10.17487/RFC2247.

[13] «Welcome to Apache Directory Studio — Apache Directory». Accedido: 30 de abril de 2024. [En línea]. Disponible en: https://directory.apache.org/studio/

[14] V. Hassler, «X.500 and LDAP security: a comparative overview», IEEE Network, vol. 13, n.º 6, pp. 54-64, dic. 1999, doi: 10.1109/65.806992.

[15] «slapd.access(5) - Linux man page». Accedido: 21 de abril de 2024. [En línea]. Disponible en: https://linux.die.net/man/5/slapd.access

[16] S. Indesteege, «Analysis and Design of Cryptographic Hash Functions».

[17] W. Eddy, «TCP SYN Flooding Attacks and Common Mitigations», Internet Engineering Task Force, Request for Comments RFC 4987, ago. 2007. doi: 10.17487/RFC4987.

[18] C. Alonso, R. Bordón, C. Alonso, y J. P. Gimeno, «LDAP Injection & Blind LDAP Injection».

[19] G. Goranov y R. Hristova, «An Approach for Integrating Joomla with LDAP», en 2019 II International Conference on High Technology for Sustainable Development (HiTech), oct. 2019, pp. 1-4. doi: 10.1109/HiTech48507.2019.9128257.

[20] «slapd(8): Stand-alone LDAP Daemon - Linux man page». Accedido: 30 de abril de 2024. [En línea]. Disponible en: https://linux.die.net/man/8/slapd

[21] «slapadd(8): Add entries to SLAPD database - Linux man page». Accedido: 30 de abril de 2024. [En línea]. Disponible en: https://linux.die.net/man/8/slapadd

[22] «LDAP/LDAPUtils - Debian Wiki». Accedido: 30 de abril de 2024. [En línea]. Disponible en: https://wiki.debian.org/LDAP/LDAPUtils

[23] «gosa-project/gosa-core». gosa-project, 22 de abril de 2024. Accedido: 1 de mayo de 2024. [En línea]. Disponible en: https://github.com/gosa-project/gosa-core

[24] «The Open Group Library». Accedido: 1 de mayo de 2024. [En línea]. Disponible en: https://publications.opengroup.org/

[25] «SSSD - System Security Services Daemon - sssd.io». Accedido: 1 de mayo de 2024. [En línea]. Disponible en: https://sssd.io/

[26] «slapo-accesslog(5) - Linux manual page». Accedido: 11 de mayo de 2024. [En línea]. Disponible en: https://man7.org/linux/man-pages/man5/slapo-accesslog.5.html

[27] «LDAP Result Code Reference», LDAP.com. Accedido: 11 de mayo de 2024. [En línea]. Disponible en: https://ldap.com/ldap-result-code-reference/

[28] R. Gerhards, «rsyslog», rsyslog. Accedido: 11 de mayo de 2024. [En línea]. Disponible en: https://www.rsyslog.com/

[29] «logrotate(8) - Linux man page». Accedido: 12 de mayo de 2024. [En línea]. Disponible en: https://linux.die.net/man/8/logrotate

[30] «cron(8): daemon to execute scheduled commands - Linux man page». Accedido: 12 de mayo de 2024. [En línea]. Disponible en: https://linux.die.net/man/8/cron

[31] «Welcome to Flask — Flask Documentation (3.0.x)». Accedido: 12 de mayo de 2024. [En línea]. Disponible en: https://flask.palletsprojects.com/en/3.0.x/

Apéndices

Apéndice A - Código

Todo el código utilizado para desarrollar la herramienta de control de acceso se encuentra disponible en el siguiente repositorio de GitHub **https://github.com/satienzaf/TFGLDAP**.

Los archivos disponibles, indicados también en el documento "README" del repositorio son:

1. Parselog.py: Código que procesa logs LDAP y genera un archivo CSV como output

2. Archivos asociados a la interfaz gráfica y REST API:

    2.1 Archivos HTML: archivos necesarios para la visualización de la interfaz web

    2.2 Run.py e init.py: archivos que ejecutan y ponen en funcionamiento el REST API

    2.3 Routes.py: archivo que contiene todos los endpoints del REST API

    2.4 Requirements.txt: archivo que contiene todas las instalaciones necesarias en Python para el funcionamiento de la API

    2.5 Styles.css: hoja de estilos de la interfaz gráfica