UNIVERSIDAD CÉSAR VALLEJO
ESCUELA DE POSTGRADO

TESIS

DISEÑO DEL PROTOCOLO SUMP PARA MEJORAR LA REVISIÓN DE LOS DERECHOS DE ACCESO DE LOS USUARIOS EN SISTEMAS OPERATIVOS LINUX DE LA EMPRESA PETROPERÚ EN LA SEDE IQUITOS - 2015

PARA OBTENER EL GRADO DE MAESTRO

EN INGENIERÍA DE SISTEMAS

CON MENCIÓN EN GESTIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN

AUTOR:

BACH. FERNANDO IGNACIO DÍAZ SÁNCHEZ

ASESOR:

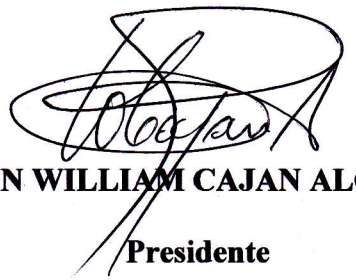
DR. OSCAR LÓPEZ REGALADO

LÍNEA DE INVESTIGACIÓN:

REDES

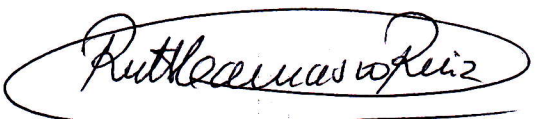
PERÚ - 2017

PAGINA DE JURADO



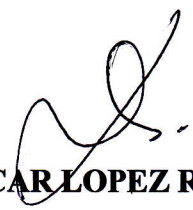
Dr. JOHN WILLIAM CAJAN ALCANTARA

Presidente



Dra. RUTH ESTHER CARRASCO RUIZ

Secretario



Dr. OSCAR LOPEZ REGALADO

Vocal

DECLARACIÓN JURADA

Yo, Fernando Ignacio Díaz Sánchez, egresado del Programa de Maestría de Ingeniería de Sistemas de la Universidad César Vallejo SAC. Chiclayo, identificado con DNI N° 41058079

DECLARO BAJO JURAMENTO QUE:

1. Soy autor de la tesis titulada: **DISEÑO DEL PROTOCOLO SUMP PARA MEJORAR LA REVISIÓN DE LOS DERECHOS DE ACCESO DE LOS USUARIOS EN SISTEMAS OPERATIVOS LINUX DE LA EMPRESA PETROPERÚ EN LA SEDE IQUITOS - 2015.**
2. La misma que presento para optar el grado de: Maestría en Ingeniería de Sistemas con Mención en Gestión de Tecnologías de la Información.
3. La tesis presentada es auténtica, siguiendo un adecuado proceso de investigación, para la cual se han respetado las normas internacionales de citas y referencias para las fuentes consultadas.
4. La tesis presentada no atenta contra derechos de terceros.
5. La tesis no ha sido publicada ni presentada anteriormente para obtener algún grado académico previo o título profesional.
6. Los datos presentados en los resultados son reales, no han sido falsificados, ni duplicados, ni copiados.

Por lo expuesto, mediante la presente asumo frente a LA UNIVERSIDAD cualquier responsabilidad que pudiera derivarse por la autoría, originalidad y veracidad del contenido de la tesis así como por los derechos sobre la obra y/o invención presentada. En consecuencia, me hago responsable frente a LA UNIVERSIDAD y frente a terceros, de cualquier daño que pudiera ocasionar a LA UNIVERSIDAD o a terceros, por el incumplimiento de lo declarado o que pudiera encontrar causa en la tesis presentada, asumiendo todas las cargas pecuniarias que pudieran derivarse de ello. Así mismo, por la presente me comprometo a asumir además todas las cargas pecuniarias que pudieran derivarse para LA UNIVERSIDAD en favor de terceros con motivo de acciones, reclamaciones o conflictos derivados del incumplimiento de lo declarado o las que encontraren causa en el contenido de la tesis.

De identificarse algún tipo de falsificación o que el trabajo de investigación haya sido publicado anteriormente; asumo las consecuencias y sanciones que de mi acción se deriven, sometiéndome a la normatividad vigente de la Universidad César Vallejo S.A.C. Chiclayo; por lo que, LA UNIVERSIDAD podrá suspender el grado y denunciar tal hecho ante las autoridades competentes, ello conforme a la Ley 27444 del Procedimiento Administrativo General.

Pimentel, 07 de Junio de 2018

Firma
Nombres y apellidos: Fernando Ignacio Díaz Sánchez
DNI: 41058079

DEDICATORIA

A Gabriela EEMM, mi bellísima esposa, te dedico este trabajo como pequeña retribución y muestra de agradecimiento a tu paciente espera, acertado apoyo y dulce compañía. Tu est et eris per semper meus magna amor.

Fernando I. Díaz Sánchez

A mi Padre Luis E. Díaz Huamán, Q.E.P.D.
por ser el artífice de haber dado inicio y fin
a esta maestría en mi carrera.
Gracias por creer en mí.

AGRADECIMIENTO

A la Universidad César Vallejo y a la Escuela de Posgrado por ser un medio para poder desarrollarme profesionalmente en una maestría de buen desempeño académico.

A los Ing. Tony Vásquez Vásquez, y Marks Navarro Yuyarima, de la sede Iquitos de la empresa Petroperú, por brindarme las facilidades para el desarrollo de mi investigación, sin su apoyo no hubiera sido posible este trabajo.

Al Dr. Oscar López Regalado, por su adecuada orientación metodológica que me ha permitido completar este ansiado proyecto.

A mi madre y hermanos, por su preocupación y apoyo en mi desarrollo profesional y personal.

El Autor

PRESENTACIÓN

Señores miembros del Jurado:

Siguiendo los reglamentos, que dicta la Escuela de Posgrado de la Universidad César Vallejo, para la producción y sustentación de Tesis de Maestría de Ingeniería de Sistemas, presento el trabajo de investigación denominado: "DISEÑO DEL PROTOCOLO SUMP PARA MEJORAR LA REVISIÓN DE LOS DERECHOS DE ACCESO DE LOS USUARIOS EN SISTEMAS OPERATIVOS LINUX DE LA EMPRESA PETROPERÚ EN LA SEDE IQUITOS - 2015"

Este trabajo constituye un aporte a la sociedad y en especial al mundo de la informática y telecomunicaciones, debido a que brinda un soporte a la gestión de usuarios y privilegios de forma sencilla y detallada, permitiendo además que cualquier investigador interesado pueda mejorar, corregir o replicar esta solución en las empresas o instituciones de su elección.

Es mi deseo y pretensión que al término de la misma, y en cumplimiento de los procedimientos estipulados en el reglamento para la elaboración y sustentación de Tesis de esta casa Superior de Estudios, pueda recibir el grado académico de Magister en Ingeniería de Sistemas con Mención en Gestión de Tecnologías de la Información.

Señores miembros del jurado espero que esta investigación sea evaluada y merezca su valiosa aprobación.

El Autor

INDICE

INDICE DE TABLAS

INDICE DE GRAFICOS

RESUMEN

La administración de los servidores Linux en la actualidad, conlleva una enorme responsabilidad debido a que sus sistemas y servicios son la base para la producción de las empresas que eligen esta plataforma. Por ello, es necesario implementar mecanismos que permitan su adecuada gestión y mejoren la seguridad de la información que almacenan.

Un aspecto crítico para esta adecuada gestión, según la norma internacional ISO/IEC 27002:2013, es el control que se debe tener en el proceso de revisión de los derechos de acceso de los usuarios así como la seguridad de sus servicios de red. Sin embargo, estas normas solo proporcionan directrices para la implementación de estos controles y no son una solución práctica en sí.

La forma común de llevar a cabo estos controles en la empresa Petroperú con su sede en Iquitos, ubicada en la Selva peruana, es la administración manual apoyada en el uso de las herramientas nativas del sistema operativo Linux que permiten realizar estas tareas. Sin embargo, esta metodología es propensa a algunos errores y toma mucho tiempo aplicarla en todos sus servidores.

Esta investigación titulada “DISEÑO DEL PROTOCOLO SUMP PARA MEJORAR LA REVISIÓN DE LOS DERECHOS DE ACCESO DE LOS USUARIOS EN SISTEMAS OPERATIVOS LINUX DE LA EMPRESA PETROPERÚ EN LA SEDE IQUITOS - 2015”, permite diseñar un nuevo protocolo llamado SUMP (Simple User Management Protocol) y sirve de guía para automatizar el proceso de revisión de los derechos de acceso de los usuarios y brindar una mejor performance así como un mejor nivel de seguridad en el tratamiento de la información.

El tipo de investigación aplicada y experimental junto con su diseño pre-experimental corresponde a un planteamiento metodológico muy importante que el investigador ha tomado en cuenta para motivar a los profesionales de las carreras afines, a las tecnologías de la información, al desarrollo de nuevos proyectos de investigación.

PALABRAS CLAVE: SUMP, ISO/IEC 27002:2013, Simple User Management Protocol, revisión de los derechos de acceso de los usuarios.

ABSTRACT

Nowadays, Linux server's administration implies an enormous responsibility because their systems and services are the foundations of the production in business which choose this platform. Therefore, it is crucial to implement mechanisms which allow their optimal management and that at the same time, improve the security regarding the information they store.

According to the international standard ISO/IEC 27002:2013, a significant aspect we must consider in order to have an adequate management, is the control over the process of reviewing user access rights, as well as the security of their network services. However, those rules and regulations provide us only with guidelines to implement these controls and they are not a practical solution per se.

In order to cope with these requirements, Petroperú-Iquitos does not count on a system; they merely manage the processes manually, they also use the native tools Linux Operating System provides which allow them to accomplish such tasks.

This research intends to design a brand new protocol, denominated SUMP (Simple User Management Protocol) and permits to automate the process of reviewing user access rights and also to contribute to improve the performance and the levels of security in the information processing.

Applied and experimental research along with its pre-experimental design correspond to a very important methodological approach which the researcher has taken into consideration in order to motivate all I.T. related careers to develop new research projects.

KEYWORDS: SUMP, ISO/IEC 27002:2013, Simple User Management Protocol, review of user access rights.

INTRODUCCIÓN

En los últimos años el mundo ha experimentado un cambio vertiginoso con el uso de las tecnologías de la información. A Julio de 2014, el número aproximado de usuarios de internet a nivel mundial era de 2,925,249,355; casi el doble del registrado en el 2008. En efecto, cada día más personas acceden a diversos servicios digitales ofrecidos en un sinfín de rubros. Desde los billones de videos en línea entregados por Youtube, pasando por las millones de comunicaciones al día vía Twitter y Facebook, hasta los sistemas core de cada negocio e institución que les permiten operar a diario.

Lógicamente, mientras más servicios son otorgados; la infraestructura necesaria para darle soporte también aumenta. Por ejemplo, en el cuarto trimestre de 2014, las ventas de servidores a nivel mundial se incrementaron un 4,8% respecto al año 2013, esto con el objetivo de continuar trabajando las 24 horas los 365 días del año.

Esta realidad tecnológica aún está lejos de detenerse. Según la empresa Cisco System, líder en el sector de redes, para el 2018 habrán casi cuatro billones de usuarios globales de internet (más del 51 por ciento de la población mundial). El promedio de ancho de banda mundial será de 42 Mbps y habrá 21 billones de dispositivos de red conectándose. (Cisco, 2015).

Detrás de todo este escenario, se encuentran los sistemas operativos que brindan las diferentes plataformas de software que hacen posible el mundo digital actual. Y dentro de los sistemas operativos para servidores más usados, encontramos al sistema Linux (derivado de Unix) que viene siendo usado en el 37% de los servidores web a nivel mundial. (W3Techs, 2017).

La revisión de los derechos de acceso de los usuarios en sistemas operativos Linux es una tarea para los encargados de TI. Que existan más usuarios en más de un servidor significa invertir más tiempo en gestionarlos y a la vez tener más cuidado en minimizar los riesgos de seguridad como los accesos no deseados. Por esta razón, en los primeros años de Linux se desarrollaron programas utilitarios que permitían gestionar a los usuarios de forma sencilla (paquetes shadow). Sin embargo, estas herramientas solían brindar una solución aislada y cada servidor Linux o servicio debía gestionar sus propios usuarios.

Con el tiempo y el creciente uso de la tecnología y las redes de comunicación, esta práctica quedó obsoleta y aparecieron nuevas formas de gestionar la información de los usuarios, entre ellas se destacó el protocolo LDAP (Lightweight Directory Access Protocol) que permitió centralizar la gestión de las cuentas de usuario y sus privilegios. De esta forma, no era necesario registrar en cada sistema o servicio la información de los usuarios y se resolvió el problema de la integridad de la información cuando se requería alterar o eliminar información.

Poco a poco los sistemas y servicios fueron incluyendo soporte para LDAP como parte de su funcionalidad y el panorama parecía ideal. Sin embargo, las desventajas no tardaron en aparecer.

Kelly C Bourne en su libro *Application Administrators Handbook: Installing, Updating and Troubleshooting Software* menciona entre otros sobre el grave riesgo de disponibilidad que se tiene al utilizar LDAP cuando, por alguna razón, este servicio deja de funcionar (todos los intentos de autenticación del usuario fallarán hasta que la base de datos centralizada esté disponible otra vez). También nos alerta sobre la complejidad de una implementación LDAP que abarque todos los servicios de una empresa.

Sean o no estas razones suficientes, lo cierto es que aún se utilizan cuentas de usuario locales, y su gestión sigue limitándose a usar los programas utilitarios como los del paquete shadow. Es común tener en varios servidores, usuarios con privilegios de administrador para cubrir situaciones de emergencia, usuarios operadores que realicen tareas nocturnas de forma manual, usuarios de servicios que no han sido configurados para interactuar con LDAP; como por ejemplo los usuarios de un motor de base de datos configurados en una aplicación web, etc.

De forma paralela, el aspecto de seguridad de la información fue cada vez más relevante, y organismos como la ISO/IEC, trabajaron en la elaboración de directrices para las normas de seguridad de la información organizacional, así como prácticas de gestión de seguridad de la información, incluyendo la selección, implementación y gestión de los controles, teniendo en cuenta el entorno de riesgo de la seguridad de la información en la organización. Estas directrices formaron parte de la *ISO/IEC 27002:2013 Information technology - Security techniques - Code of practice for information security controls,* la cual proporciona guías sobre la gestión de usuarios y el tratamiento de la seguridad en

base a la encriptación de los datos. Estos puntos resultan ser cruciales para solucionar el problema abordado previamente.

El presente informe, constituye un esfuerzo por brindar una propuesta alternativa a la empresa Petroperú en la sede Iquitos, al problema de la centralización en la gestión de cuentas de usuario locales en sistemas operativos Linux, y en concreto del proceso de revisión de sus derechos de acceso siguiendo las guías establecidas en los controles 9.2.5 y 13.1.2 de la ISO/IEC 27002:2013, mediante el diseño de un nuevo protocolo denominado SUMP (Simple User Management Protocol).

El proyecto pretende cumplir con 4 objetivos específicos para hacer posible la propuesta planteada, los cuales se resumen a continuación: a) Identificar los diferentes sistemas y servicios utilizados en los servidores de Petroperú de la sede Iquitos que requieren de un protocolo de gestión de usuarios. b) Determinar el estado actual del proceso de revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en su sede Iquitos. c) Diseñar el protocolo SUMP para mejorar la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux. d) Evaluar la simulación del protocolo SUMP en la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux.

Esta investigación consiste de 5 capítulos los cuales están estructurados de la siguiente forma:

El capítulo I, abarca el problema de la investigación, se detalla la situación actual de la empresa Petroperú sede Iquitos en relación al tema de investigación; así mismo se plantean los objetivos a cumplir.

El capítulo II, abarca el marco teórico y conceptual, donde se presenta el sustento científico y se recopila información sobre las variables, los términos y definiciones que van a apoyar a la explicación del proyecto de investigación.

El capítulo III, comprende el marco metodológico, que incluye la hipótesis, definición de variables, metodología, métodos, la población y muestra que se usará en la investigación así como las técnicas e instrumentos de recolección de datos.

El capítulo IV, mostrará los resultados de la investigación con su respectivo análisis e interpretación por cada objetivo planteado.

El capítulo V, entregará las conclusiones y sugerencias relacionadas con el proyecto de investigación que se derivan de los resultados obtenidos.

CAPITULO I
PROBLEMA DE INVESTIGACIÓN

1. CAPITULO I - PROBLEMA DE INVESTIGACION

1.1. Planteamiento del problema

Con los años, el crecimiento de la tecnología a nivel mundial ha sido abrumador y ha conseguido atraer a más personas mediante diferentes servicios (Youtube, Facebook, Twitter, etc.). Para que toda esta tecnología pueda llegar al público se necesita de un punto de ingreso hacia ella, y se requiere que los servicios sean capaces de diferenciar a una persona (o dispositivo) de otra. No solo eso, también es importante tener información de las actividades que puede realizar una persona en un determinado servicio.

En el Perú, trabajar con usuarios y sus privilegios son actividades comunes de las personas a cargo de administrar los servidores y servicios de TI, y hacerlo no es una tarea sencilla aunque lo parezca. Requiere tiempo y mucha atención para poder manipular información relacionada con los usuarios. Al ser una labor delicada, un mínimo error puede producir el corte parcial o total del servicio hacia un cliente, y si se trabajan con muchos servidores y servicios, el control de estos usuarios se vuelve tedioso rápidamente.

Por otro lado, los servidores y servicios suelen venir por separado y cada servicio ofrece su propio mecanismo de trabajar con usuarios y privilegios (el soporte LDAP no suele ser el método de autenticación por defecto). Por ejemplo, los servidores Linux controlan el acceso de los usuarios de forma distinta que el servicio MySQL que sirve para gestionar bases de datos; incluso si ambos trabajan bajo el mismo hardware. Por lo tanto, atender estos requerimientos suele tomar mucho tiempo si se realiza de forma manual.

Este problema no es ajeno a las empresas locales, y en concreto a Petroperú con su sede en la región Selva (Iquitos) en donde, si bien, se mantiene trabajando con estándares ITIL para el buen gobierno de la tecnología, incurre de forma frecuente en desvíos a sus propios procedimientos de seguridad y entre ellos la relacionada con la gestión de usuarios y privilegios.

Si bien es cierto, Petroperú utiliza servidores LDAP como parte de su infraestructura de red, la realidad es que en todos sus servidores también utiliza cuentas de usuario locales que las utilizan los administradores de red y los operadores, además, los servicios

como bases de datos web no están configurados con soporte LDAP lo que conlleva a su gestión manual.

Esta realidad surge en gran medida por la falta de mecanismos automáticos que centralicen la gestión de usuarios y sus privilegios; Petroperú, a pesar de tener identificado sus servidores y servicios y de contar con guías para obtener la información necesaria sobre sus usuarios, no controla adecuadamente sus accesos. Se pierde el control al dejar a criterio de cada administrador de los servicios la forma de lidiar con este problema, un ejemplo recurrente de estos problemas son *los permisos espurios* que quedan en los sistemas tras el cese de algún personal y cuya información aún permanece en ellos.

Como consecuencia, la empresa Petroperú en su sede Iquitos tiene un control de usuarios que debe mejorar para ahorrar tiempo y dinero, por tanto se evidencia la necesidad de contar con un mecanismo que permita brindar una solución al problema de la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux.

1.2. Formulación del problema

¿En qué medida el protocolo SUMP mejora la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en la sede Iquitos - 2015?

1.3. Justificación

Justificación Teórica

Para König (2012), nuestra experiencia cotidiana ‘casi libre de errores’ al usar los servicios de Internet, esconde el enorme esfuerzo requerido para el desarrollo de los protocolos de comunicación. Esta reflexión nos exige estudiar el camino que siguen los protocolos antes de ser utilizados, de tal manera que se pueda descubrir, comprender y mejorar sus diseños. Así mismo, mediante esta investigación se permitirá contrastar los resultados obtenidos al seguir el paradigma de diseño de protocolos y presentar una solución al problema de gestión centralizada de usuarios y privilegios de forma segura.

Justificación Práctica

Según la ISO/IEC (2013), en su estándar internacional ISO/IEC 27002:2013, existe la necesidad de establecer controles que permitan revisar los derechos de acceso de los usuarios para mejorar la gestión de seguridad de la información. Estos controles son realizados de forma manual en la sede Iquitos de Petroperú en todos sus servidores, y desafortunadamente esta metodología de trabajo presenta algunos errores y su aplicación toma mucho tiempo.

Como respuesta a la inquietud de proporcionar un mecanismo que permita implementar de forma eficiente y segura las recomendaciones entregadas por el estándar internacional ISO/IEC 27002:2013, se debe evaluar si el diseño de un nuevo protocolo mejora esta actividad.

Justificación Social

Esta investigación representa un pequeño aporte social en el campo tecnológico de redes y comunicaciones, porque permite definir un nuevo protocolo que mejora la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux, permitiendo una gestión de usuarios de forma centralizada y optimizando el tiempo invertido en la atención de este proceso. Así mismo es importante considerar que la presente investigación servirá como modelo de referencia para el planteamiento de otras investigaciones en el campo de redes que puedan realizar otras instituciones.

1.4. Antecedentes

En el Mundo

Barros Arteaga, (2013), realizó la investigación “Diseño de Protocolos para redes inalámbricas de sensores sobre LatinOS: Metodología y Ejemplos” en la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile. Su investigación llegó a estas conclusiones relevantes para este proyecto de investigación:

1. Es muy importante la elaboración de una metodología para el desarrollo de un protocolo que sea verificada mediante aplicaciones de ejemplo y desplegada en el terreno tecnológico.
2. La metodología debe ser versátil para poder desarrollar protocolos que funcionen de forma modular.

3. La implementación debe contemplar la portabilidad entre sistemas operativos basándose en sus especificaciones técnicas.

Estas conclusiones son una guía inicial sobre como esbozar con mayor claridad algunos aspectos centrales de esta investigación, como la elección de un diseño modular durante su fase de diseño.

Butcher, (2007), en su libro “Mastering OpenLDAP” describe que el protocolo LDAP (Protocolo Ligero de Acceso al Directorio) “fue diseñado originalmente para ser un protocolo que provea una forma alternativa de acceso a los servidores de directorio existentes” (Butcher, 2007, p. 7).

Esta literatura, aporta las bases para la construcción de un mecanismo de consulta de cuentas de usuarios y privilegios mediante su almacenamiento en directorios. Este protocolo sin embargo, no contempla una arquitectura distribuida sino más bien centralizada de tal forma que no se ocupa de las cuentas locales de cada sistema o servicio. Por otro lado, “LDAP es estandarizado. el contenido del estándar LDAP, incluyendo los protocolos de red, la estructura de directorio, y los servicios otorgados por un servidor LDAP, están todos disponibles en la forma de RFCs (Request For Comments)” (Butcher, 2007, p. 7). Este modelo de trabajo ha tenido una influencia fundamental que se ha elegido para el diseño del protocolo SUMP.

Mauro & Schmidt, (2005), en su libro “Essential SNMP” sostiene que el Protocolo Simple de Administración de Red (SNMP) es un protocolo de la capa de aplicación definido como “un conjunto simple de operaciones (y la información obtenida) que da a los administradores la habilidad de cambiar el estado de algunos dispositivos basados en SNMP” (Mauro & Schmidt, 2005, p. 1).

El aporte que brinda este autor a esta investigación, es la explicación técnica sobre el funcionamiento interno del protocolo SNMP, que sirve de guía de diseño para la transmisión de datos mediante la simplificación de peticiones y respuestas de tal manera que se reduce la complejidad en su uso, diseño y programación.

En el Perú

En la empresa Petroperú al año 2015, el servicio de tecnologías de la información ha sido gestionado por la empresa transnacional IBM (International Business Machines), que utiliza software propietario para las distintas actividades tecnológicas. Entre sus herramientas, existe una denominada UAT (User Administration Tool) que permite determinar en cualquier momento lo siguiente:

- Cuáles son los usuarios compartidos en un sistema operativo
- Donde se almacenan los usuarios y claves
- Quien está utilizando la clave de un usuario

La relación con la investigación es muy estrecha, puesto que este software ha sido la inspiración para la elaboración del protocolo SUMP. Las diferencias sin embargo con este producto son notorias; en primer lugar el software UAT cubre un mayor alcance en relación a la gestión de usuarios (control de solicitudes de claves, gestión de aprobaciones de solicitudes, manipulación de claves de usuario), en cambio el protocolo SUMP solo está centrado en la revisión de usuarios y privilegios, de tal forma que la rendimiento sea mucho mayor.

El software UAT es una suite de herramientas que incluye un panel de administración web; sin embargo el protocolo SUMP solo se ocupa de la transmisión de la información más importante (los usuarios, grupos y privilegios) y deja a terceros la construcción de herramientas de administración web. El software UAT es propiedad de IBM sin posibilidad de revisar su código fuente o su diseño interno, y el protocolo SUMP se diseña por completo en este trabajo de investigación, incluyendo su código fuente.

1.5. Limitaciones

La información relacionada con el diseño e implementación de un protocolo es limitada y antigua. Para poder completar este trabajo de investigación se tuvo que recurrir a información relacionada con sistemas operativos y descripción de protocolos ya existentes en las RFCs (Request For Comments).

1.6. Objetivos

1.6.1. Objetivo general

Diseñar el protocolo SUMP para mejorar significativamente la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en la sede Iquitos - 2015.

1.6.2. Objetivos específicos

1. Identificar los diferentes sistemas y servicios utilizados en los servidores de Petroperú en su sede Iquitos que requieren de un protocolo de gestión de usuarios.
2. Determinar el estado actual del proceso de revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en su sede Iquitos.
3. Diseñar el protocolo SUMP para mejorar la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux.
4. Evaluar la simulación del protocolo SUMP en la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en la sede Iquitos.

CAPITULO II
MARCO TEÓRICO - CONCEPTUAL

2. CAPITULO II - MARCO TEORICO Y CONCEPTUAL

2.1. Marco Teórico

2.1.1. Protocolo SUMP

2.1.1.1. Redes de computadoras

Una red de computadoras, según (Tanenbaum & Wetherall, 2011, p. 2) es un conjunto de computadoras autónomas interconectadas, cuya conexión se realiza no solo mediante un cable de cobre; sino también se pueden utilizar las fibras ópticas, las microondas, los rayos infrarrojos y los satélites de comunicaciones.

Lo que lo diferencia de otras redes (como la red telefónica o la red de cable) según (Peterson & Davie, 2012, p. 2) es su generalidad, pues las redes de computadoras están construidas desde un hardware programable de propósito general y no están optimizados por una aplicación en particular como realizar llamadas telefónicas o brindar señales de televisión. En su lugar, permiten llevar diferentes tipos de datos, y soportar un amplio y creciente rango de aplicaciones.

2.1.1.1.1. Usos de las redes de computadoras

Para (Tanenbaum & Wetherall, 2011, p. 3), el uso de las redes de computadoras se clasifica según el interés de las personas y del uso que le pueden dar. Así pues se pueden anotar las aplicaciones en los negocios, en el hogar, los usuarios móviles y los temas sociales.

2.1.1.1.1.1. Aplicaciones de Negocios

En los negocios, (Tanenbaum & Wetherall, 2011, pp. 3-4) afirma que, la compartición de recursos es el asunto principal donde “el objetivo es hacer que todos los programas, el equipo y, en particular, los datos estén disponibles para todos los que se conecten a la red, independientemente de la ubicación física del recurso y del usuario”.

Así mismo, la información se almacena en equipos poderosos llamados servidores que son utilizados por los empleados desde equipos clientes, formando el denominado modelo cliente-servidor.

El segundo objetivo tiene que ver con las personas y se trata de la comunicación que deben establecer los empleados. Por último el tercer objetivo es poder realizar negocios de manera electrónica con proveedores y clientes.

Entre las aplicaciones más resaltantes en la empresa tenemos el correo electrónico, la telefonía IP o VoIP, el acceso a escritorio remoto, la conexión VPN y el comercio electrónico.

2.1.1.1.1.2. Aplicaciones para el Hogar

Las personas hoy en día compran computadoras para conectarlas a internet y es, según sostienen (Tanenbaum & Wetherall, 2011, pp. 6-9)**,** para poder sacar provecho del acceso a la información remota mediante la WWW, la mensajería instantánea mediante el chat, entretenimiento interactivo como juegos o videos en línea y comercio electrónico.

2.1.1.1.1.3. Usuarios Móviles

También nos explica (Tanenbaum & Wetherall, 2011, pp. 9-10)**,** que los usuarios que tienen dispositivos móviles quieren realizar todas las tareas que realizan en sus hogares (revisar emails, ver películas, navegar en la web, etc.) pero desde cualquier ubicación.

Las redes inalámbricas son de gran interés puesto que permite conectarse a internet con un simple punto de acceso (basado en el estándar 802.11) de forma sencilla y rápida. Así mismo, la convergencia entre las redes inalámbricas y la telefonía ha permitido que aplicaciones como la ubicación GPS y los mensajes de texto sean de gran popularidad (Tanenbaum & Wetherall, 2011).

2.1.1.1.1.4. Temas sociales

Para (Tanenbaum & Wetherall, 2011, pp. 14-16)**,** las redes sociales, los sitios para compartir contenidos y otras aplicaciones han permitido a las personas compartir su forma de pensar con otros individuos con intereses similares. Las redes de computadoras hacen muy sencilla la comunicación pero también crean conflictos debido a la facilidad para espiar dichas comunicaciones.

Por otro lado, las redes de computadoras son muy buenas para incrementar la seguridad con el envío de mensajes anónimos o encriptados y evitar el seguimiento mediante cookies en los navegadores o evitar el spam y el phishing provenientes de un botnet.

2.1.1.1.2. Clasificación de redes

En la clasificación de las redes, (Tanenbaum & Wetherall, 2011, p. 3) refieren que no hay una sola clasificación aceptada para las redes de computadoras, sin embargo menciona que 2 de ellas destacan de forma importante.

2.1.1.1.2.1. Tecnología de transmisión

2.1.1.1.2.1.1. Enlaces de difusión (broadcast)

Son aquellas redes con un solo canal de comunicación en donde todos los mensajes enviados a dicha red son recibidos por todos sus miembros, quienes revisan en los paquetes la dirección de destino para determinar si han sido enviados para ellos o no y de esta forma procesarlos o eliminarlos según sea el caso. Un ejemplo de enlace de difusión son las redes inalámbricas (Tanenbaum & Wetherall, 2011).

2.1.1.1.2.1.2. Enlaces de multidifusión

Los sistemas de difusión permiten un comportamiento singular en el cual se puede enviar los mensajes de tal forma que todos sus miembros puedan procesarlos (como si todos fueran el destinatario). Este modo de operar es conocido como broadcasting. Así mismo, también se puede "destinar" los mensajes solo a un subconjunto de los miembros de la red, lo cual es conocido como multicasting. (Tanenbaum & Wetherall, 2011).

Un ejemplo de aplicación de multicasting es la videoconferencia que permite a un grupo de computadoras conectadas a la red intercambiar voz y video de forma simultánea (Halsall, 2005).

2.1.1.1.2.1.3. Enlaces punto a punto

Los enlaces punto a punto, manifiestan (Tanenbaum & Wetherall, 2011), son aquellos que establecen muchas conexiones entre pares individuales de máquinas. Para que los mensajes lleguen a su destino es

posible que se utilicen varias máquinas intermedias, por lo que es importante que en estas redes se busque la eficiencia en las rutas a seguir para llegar a su destino.

2.1.1.1.2.2. Alcance de la red

2.1.1.1.2.2.1. Redes de área personal (PAN)

Son redes destinadas a una sola persona para ser usada en la comunicación de sus dispositivos inalámbricos de corto alcance como el mouse y teclado bluetooth o dispositivos RFID como smartcards (Tanenbaum & Wetherall, 2011).

2.1.1.1.2.2.2. Redes de área local (LAN)

Son redes privadas generalmente ubicadas en un solo edificio o campus de pocos kilómetros, usadas para conectar estaciones de trabajo para compartir recursos como impresoras o archivos. Estas redes pueden trabajar en diferentes medios de transmisión (cableado o inalámbrico) así como en diferentes topologías siendo la topología Ethernet la más utilizada (Tanenbaum & Wetherall, 2011).

2.1.1.1.2.2.3. Redes de área metropolitana (MAN)

Son redes que cubren una ciudad entera, siendo la red de televisión por cable la más conocida. Con el surgimiento de Internet la red de televisión por cable fue adaptada para proporcionar a las computadoras de los hogares el acceso a la red de redes. Más adelante se desarrollaría la tecnología WiMAX que es otra forma de red MAN de alta velocidad pero con acceso inalámbrico (Tanenbaum & Wetherall, 2011).

2.1.1.1.2.2.4. Redes de área amplia (WAN)

Son redes de gran alcance geográfico como países o continentes las cuales tienen como función comunicar computadoras (hosts) mediante el uso de enrutadores (routers) y líneas de transmisión dentro de una subred de servicio privado como proveedores de Internet (ISP) o compañías telefónicas (Tanenbaum & Wetherall, 2011).

Aunque tenga similitud con la red LAN, las diferencias con la red WAN es que los hosts y las subredes tienen distintos propietarios y operadores,

además, los enrutadores permiten conectar diferentes tecnologías de red y de diferentes tamaños como computadoras individuales o redes LAN enteras (Tanenbaum & Wetherall, 2011).

Algunos ejemplos de redes WAN son las redes satelitales de TV, la red celular, y las redes VPN (Tanenbaum & Wetherall, 2011).

2.1.1.1.2.2.5. Interredes

Las interredes según (Tanenbaum & Wetherall, 2011), son una colección interconectada de redes diferentes. La Internet es la interred más grande que utiliza las redes ISP para conectar empresas, hogares y muchas otras redes. Sin embargo una interred puede ser también la conexión de 2 redes LAN unidas por una WAN. Actualmente no hay mucho consenso sobre la terminología adecuada de las interredes pero se sugiere utilizar dicha denominación cuando se unen 2 redes de distintos propietarios y que usan diferentes tecnologías.

2.1.1.2. Protocolos

Un protocolo es un conjunto de reglas o acuerdos sobre el intercambio de información que se dará lugar entre dos programas que desean comunicarse en una red de computadoras. Estas reglas indican por ejemplo como están estructurados los paquetes, en donde se guarda la información o cuán grande es. El diseño de un protocolo tiene por finalidad resolver un problema concreto haciendo uso de sus capacidades, por ejemplo el protocolo HTTP que resuelve el problema de transferir hipertexto entre servidores web y navegadores (Makofske, Donahoo, & Calvert, 2004).

Por su parte, (König, 2012) refiere que las reglas que se siguen durante la interacción entre un par de entidades son definidas por un protocolo de comunicación o simplemente un protocolo, cuyo comportamiento es una convención que define el orden temporal de las interacciones entre el par de entidades así como el formato (sintáctico y semántico) de los mensajes intercambiados.

Así mismo, (Sharp, 2008) sostiene que un protocolo es la comunicación entre computadoras que se lleva a cabo por el intercambio de datos, donde el

intercambio se da en pasos discretos denominados “elementos de comunicación” y cada uno de ellos transfiere un mensaje. El intercambio de datos puedes darse entre dos o más partes denominándose “comunicación N-peer” al intercambio entre N partes y “protocolo N-peer” al protocolo que lo permite.

En una comunicación “N-peer” el protocolo define un lenguaje para cada una de las N partes cuyas sentencias son secuencias de mensajes legales recibidas por esa parte y cuyo alfabeto de símbolos son un subconjunto de todos los mensajes posibles. Para que una computadora pueda obedecer las reglas del protocolo, debe poder reconocer dicho lenguaje (Sharp, 2008).

2.1.1.2.1. Capas y protocolos

Los protocolos son ampliamente utilizados en las redes de computadoras, las cuales fueron diseñadas por capas para reducir su complejidad. Así, cuando se lleva a cabo una comunicación entre máquinas, cada capa se comunica con su capa par y las reglas que rigen dicha comunicación se denomina protocolo de capa N (Tanenbaum & Wetherall, 2011).

El diseño por capas brinda dos ventajas: descomponer el problema de construir una red en componentes más manejables en lugar de construir una sola pieza de software y la ventaja de proveer un diseño modular que permite reusar la funcionalidad de cada capa (Peterson & Davie, 2012).

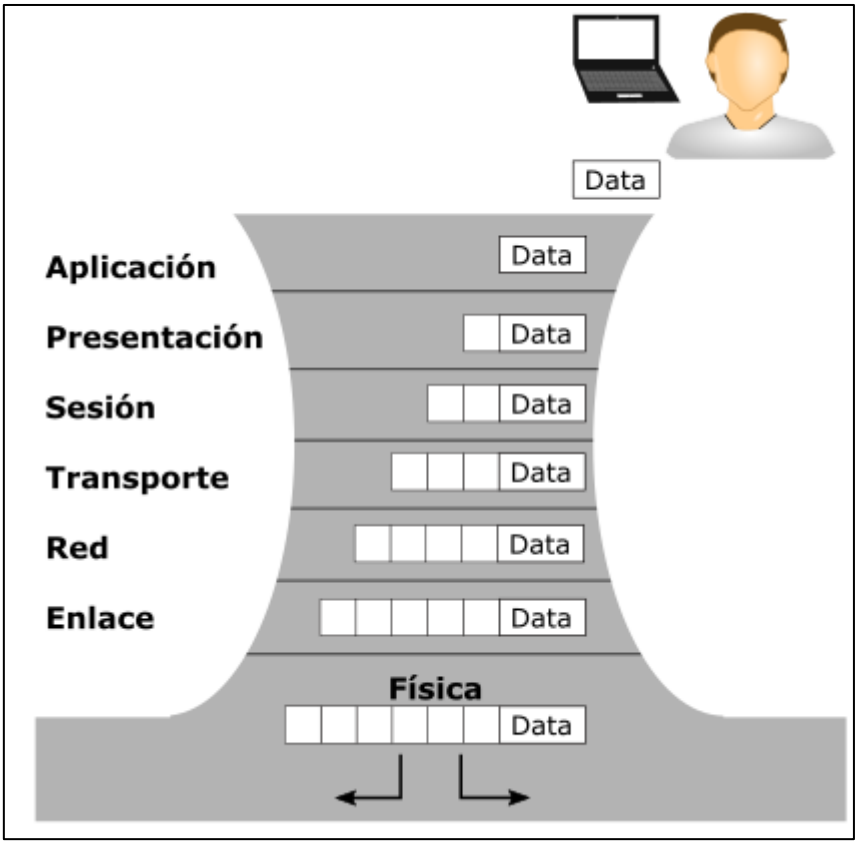
El conjunto de capas y protocolos se denomina arquitectura de red, mientras que un conjunto de protocolos (uno por capa) usados por ciertos sistemas se denomina pila de protocolos. Dos arquitecturas de red son las más importantes y se les denomina comúnmente modelos de referencia (Tanenbaum & Wetherall, 2011).

2.1.1.2.2. Modelo de referencia OSI

En 1977 la ISO (International Organization for Standardization) adoptó el modelo OSI (Open Standards Interconnection) como respuesta a la complejidad que representaba las comunicaciones de red; para ello separó las tareas involucradas en mover datos de un host a otro (Blank, 2004).

El modelo de referencia OSI está dividido en 7 capas en donde cada capa desempeña un rol en crear un paquete de datos con información crítica que describe los datos (encapsulación), finalmente este paquete es enviado de un dispositivo a otro donde será tomado y leído (des-encapsulación) (McMillan, 2012).

Gráfico 1: Las 7 capas del Modelo OSI



Fuente: (Blank, 2004, p. 20)

2.1.1.2.2.1. La capa física

En la capa física, (Blank, 2004) indica que se determina como los bits de datos se envían y reciben a lo largo de la red, trabajando así al nivel más bajo de todos poniendo y sacando datos de los cables.

Por su parte, (Tanenbaum & Wetherall, 2011), refieren que la capa física está pendiente de la transmisión de bits sobre un canal de comunicación, lidiando por diseño con la garantía en la entrega y recepción de cada bit. Los dilemas de diseño en esta capa tienen que ver con aspectos

mecánicos, eléctricos y de temporización, por ejemplo que señal eléctrica usar para representar los bits o cuantos nanosegundos dura un bit, como empieza o termina las conexiones.

Queda claro que los medios de transmisión no se limitan a los cables de cobre sino a cualquier medio que haga posible la creación de un canal de comunicación. Según lo que explica (McMillan, 2012), si se trata de una red cableada, los bits se representarán con la presencia o ausencia de cargas eléctricas, si se trata de una red inalámbrica se usarán ondas de radio alteradas o moduladas para poder diferenciar los unos de los ceros. Si el cable es de fibra óptica se usarán patrones de luz generados por pequeños láseres que se encienden y se apagan para indicar la presencia de unos y ceros.

2.1.1.2.2.2. La capa de enlace de datos

La capa de enlace de datos tal como lo refiere (McMillan, 2012), se encarga de convertir los identificadores lógicos de la capa de red como una dirección IP en identificadores físicos, por ejemplo el identificador utilizado por el protocolo Ethernet es la dirección MAC (Media Access Control) o el DLCI (Data-Link Connection Identifiers) es el identificador cuando se trata del protocolo WAN Frame Relay.

Aquí es donde los datos son preparados para su envío final hacia la red. El paquete recibido de la capa de red es encapsulado en tramas (frames) y los protocolos de esta capa ayudan en la detección de errores y el direccionamiento (identificación) de los datos que serán enviados (Blank, 2004).

2.1.1.2.2.3. La capa de red

Esta capa es responsable del enrutamiento de paquetes mediante sus direcciones lógicas. Esta capa fragmenta y ensambla paquetes y los traslado a diferentes redes si es necesario (Blank, 2004).

Los estudios de (Tanenbaum & Wetherall, 2011), nos refieren a que en esta capa debe determinarse como enrutar los paquetes desde su origen a su destino y puede realizarse mediante tablas de enrutamiento estático.

También tiene la responsabilidad de controlar la congestión, la calidad del servicio y la interconexión entre redes.

Como información adicional, (McMillan, 2012) aclara que la identificación lógica que se da en la capa de red es llamada dirección IP que no es más que un número único que diferencia a un host de los demás dispositivos en la red. Su numeración está basada en un sistema numérico que permite conocer si un equipo está o no en su misma red.

2.1.1.2.2.4. La capa de transporte

La capa de transporte según (McMillan, 2012), permite a las computadoras de destino identificar la aplicación (o servicio) que está siendo utilizada. Lo consigue usando un número de puerto. Los números de puertos van desde 1 hasta 65535, de los cuales los primeros 1023 (denominados puertos bien conocidos) ya han sido estandarizados.

Los 2 protocolos que operan en la capa de transporte son el protocolo TCP (Transmission Control Protocol) y el UDP (User Datagram Protocol) y cada uno controla sus propios tipos de numeración para los puertos; el uso de uno u otro protocolo depende del tipo de entrega que se desea (McMillan, 2012).

Por su parte, (Tanenbaum & Wetherall, 2011) refieren que la función básica de esta capa es dividir los datos de las capas superiores en unidades más pequeñas y enviarlas a la capa de red asegurando que todas las piezas lleguen a su destino de forma eficiente y transparente incluso frente a cambios de tecnología de hardware.

2.1.1.2.2.5. La capa de sesión

Esta capa configura sesiones y controla el diálogo que se da durante las comunicaciones. Los protocolos de esta capa se ocupan de cómo establecer una conexión, como usarla, como cerrarla cuando la sesión termina y revisar errores en la transmisión de datos (Blank, 2004).

Por otra parte, (Tanenbaum & Wetherall, 2011) sostienen que esta capa permite a los usuarios establecer sesiones entre ellos en equipos

diferentes, permitiendo el control de diálogo (a quien le toca transmitir) así como la administración de token (para impedir realizar ciertas acciones al mismo tiempo) y la sincronización como reinicios de sesión.

2.1.1.2.2.6. La capa de presentación

Esta capa es responsable por la manera en que los datos son representados desde el origen hasta el equipo de destino y si es necesario algún tipo de traducción esta capa debe hacerlo. Entre algunos ejemplos tenemos al uso de datos encriptados o al uso de datos comprimidos que deben presentarse adecuadamente al destinatario para que lo entienda (McMillan, 2012).

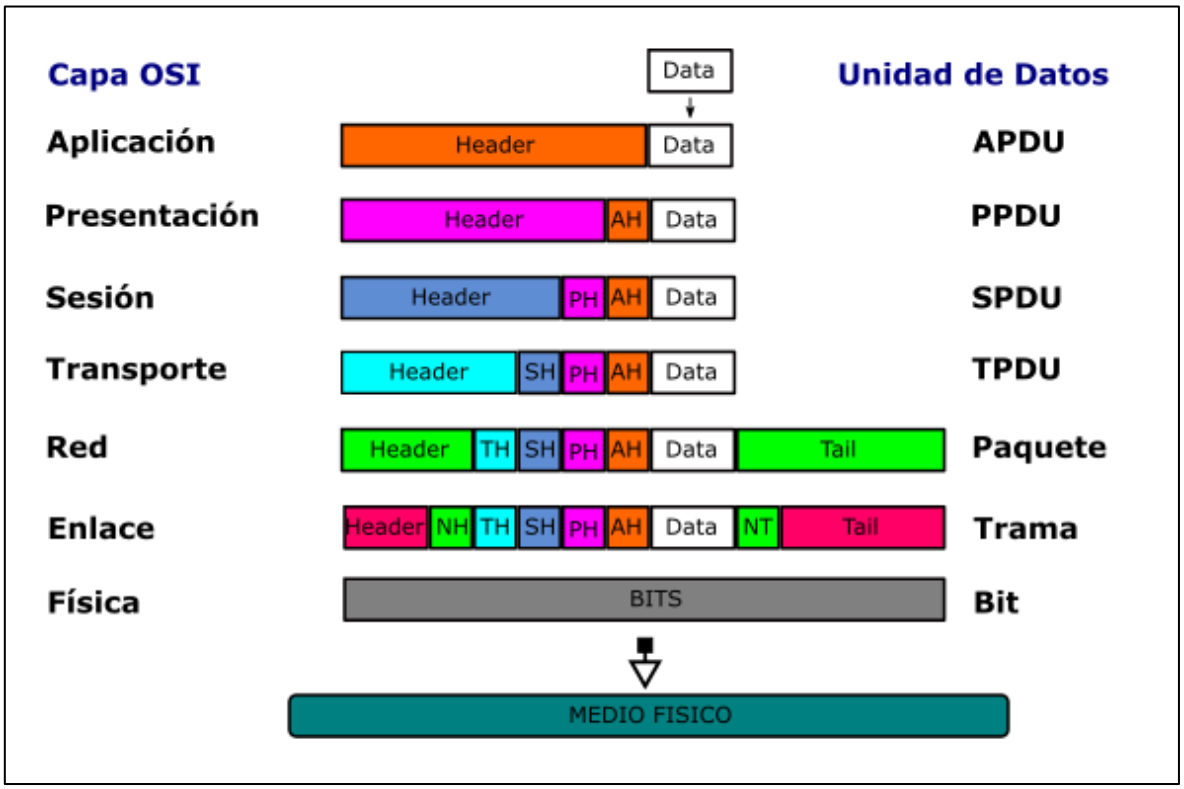
La función principal de esta capa según (Blank, 2004), es asegurar que los mensajes sean transmitidos en un lenguaje o sintaxis que el destinatario pueda entender y para ello es probable que recurra a algún mecanismo de traducción.

2.1.1.2.2.7. La capa de aplicación

El propósito de esta capa es gestionar las comunicaciones entre las aplicaciones que reciben y solicitan los datos; podemos apreciarla como el CEO del modelo OSI (Blank, 2004).

Los datos de las aplicaciones existen en esta capa y pueden ser documentos de cualquier tipo como HTML, texto plano, hojas de cálculo, documentos de word o documentos pdf. Algunos protocolos de esta capa tenemos al protocolo HTTP, DNS, FTP o SMTP (McMillan, 2012).

Gráfico 2: Formato de los datos de la Pila OSI



Fuente: (Wikipedia, 2017)

2.1.1.2.3. Modelo de referencia TCP/IP

En 1969 el departamento de defensa de EEUU a travez de su agencia ARPA y la colaboración de algunas universidades creó la red ARPANET que funcionaba con el protocolo NCP (Network Control Protocol) conectando varias redes, sin embargo al notar que este protocolo no era escalable lo cambiaron por la suite de protocolos TCP/IP (McMillan, 2012).

Este modelo surgió según (Tanenbaum & Wetherall, 2011) por la necesidad de agencia ARPA de tener una nueva arquitectura, y diseño un nuevo modelo de referencia que permitiera conectar varias redes de manera sólida y que pudiera sobrevivir a la pérdida de hardware de la subred sin que las comunicaciones se interrumpan. Este nuevo modelo lo llamo modelo de referencia TCP/IP.

En este modelo, existen 4 capas que se detallan a continuación.

2.1.1.2.3.1. La capa de enlace

Esta capa describe lo que los diferentes enlaces (como líneas seriales o ethernet) deben hacer para conocer los requerimientos de la capa de internet. Más que una capa es una interface entre los host y los enlaces de transmisión (Tanenbaum & Wetherall, 2011).

Comparándolo con el modelo OSI, la capa de enlace del modelo TCP/IP engloba a la capa física y la capa de enlace (McMillan, 2012).

2.1.1.2.3.2. La capa de internet

Esta capa es la que sostiene a toda la arquitectura ya que permite que los host inyecten paquetes a cualquier red y dejar que viajen de forma independiente aunque sea a redes diferentes. Incluso los paquetes pueden llegar desordenados y de ser el caso enviarlos a capas superiores para su reordenamiento. Su capa correspondiente en el modelo OSI es la capa de red (Tanenbaum & Wetherall, 2011).

2.1.1.2.3.3. La capa de transporte

En la capa de transporte, (McMillan, 2012) afirma que las partes de la suite TCP/IP se dedican a usar la información suministradas por los números de puertos. Los protocolos UDP y TCP operan en esta capa.

El protocolo TCP, es un protocolo confiable y orientado a la conexión, garantizando la entrega sin errores a su máquina de destino en la red. Lo consigue dividiendo el flujo de bytes en mensajes discretos que luego re ensambla llegado a su destino; incluso puede controlar el flujo de datos para no saturar al receptor (Tanenbaum & Wetherall, 2011).

En cambio el protocolo UDP según (Tanenbaum & Wetherall, 2011), no es confiable ni orientado a conexión y lo utilizan las aplicaciones que no desean un secuencia dada o un control de flujo ya que desean emplear los suyos. Es muy usada en aplicaciones como voz o video donde lo más importante es que la información llegue a tiempo sin importar cuan precisa sea.

2.1.1.2.3.4. La capa de aplicación

La capa de aplicación, tal como lo explican (Tanenbaum & Wetherall, 2011), contiene los protocolos de alto nivel como HTTP, SMTP, DNS, etc. El modelo TCP/IP carece de capas de sesión y presentación, y en su lugar las aplicaciones incluyen esas funcionalidades según la requieran.

Por su parte, (McMillan, 2012) sostiene que, esta capa es la interfaz que usan las aplicaciones para acceder a los procesos de red que facilita la suite TCP/IP, de tal forma que cuando un usuario desea obtener un recurso que se encuentra en la red, esta capa comienza el proceso de crear los paquetes que se utilizarán para pedir dicho recurso al dispositivo remoto que lo tenga.

2.1.1.2.4. Diseño de protocolos

Los protocolos al igual que otro tipo de sistemas, son piezas de software que siguen las instrucciones para el cual fueron diseñados, con el objetivo de resolver algún problema. Por lo tanto, comparten una de principales fases del desarrollo de software, que es precisamente el diseño.

El diseño de software “visto como un proceso, es la actividad del ciclo de vida del software donde se analizan los requerimientos para producir una descripción de la estructura interna del software que servirá de base para su construcción”. (IEEE Computer Society, 2014, p. 50)

Por su parte, la (IEEE Computer Society, 2014) precisa que el diseño de software consiste en 2 procesos relacionados. Primero está el diseño de la arquitectura donde se definen los componentes e interfaces del sistema y segundo está el diseño detallado el cual describe el comportamiento de estos elementos. El resultado o salida de estos procesos será un conjunto de artefactos y modelos donde se reflejarán las decisiones críticas que han sido tomadas en el camino.

Si bien es cierto los protocolos son piezas de software, (König, 2012) argumenta que, su diseño y desarrollo difieren del software tradicional en algunos aspectos. En primer lugar se trabajan con especificaciones (de servicio, de protocolo y de implementación) en lugar de una lista de

requerimientos fija durante el análisis. Así mismo, un protocolo debe ser diseñado pensando en diferentes sistemas de cómputo con varios sistemas operativos por lo que la abstracción es muy importante. En segundo lugar, la independencia de plataforma fuerza a que se dejen ciertas decisiones abiertas en la descripción del protocolo y se resuelvan en la implementación. Finalmente, el diseño de protocolos necesita utilizar una descripción estándar internacional para conseguir la independencia en su implementación.

La fase de diseño en el proceso de desarrollo de un protocolo según (König, 2012), contiene 3 fases: el diseño del servicio, el diseño del protocolo y el diseño de implementación; sin embargo, la metodología utilizada para abordar estas fases pueden ser formales o no formales (heurísticas) debido a que no existen métodos o técnicas aceptadas a nivel mundial.

2.1.1.2.4.1. Metodología de diseño formal

En su investigación, (König, 2012) nos presenta una metodología formal para la etapa de diseño de protocolos que cubre sus 3 fases de diseño (servicio, protocolo e implementación) basadas en la construcción de especificaciones mediante descripciones formales que documentan las decisiones del diseño. El uso de técnicas de descripción formal (FDTs) durante la fase de diseño se beneficia de un desarrollo iterativo en cada especificación.

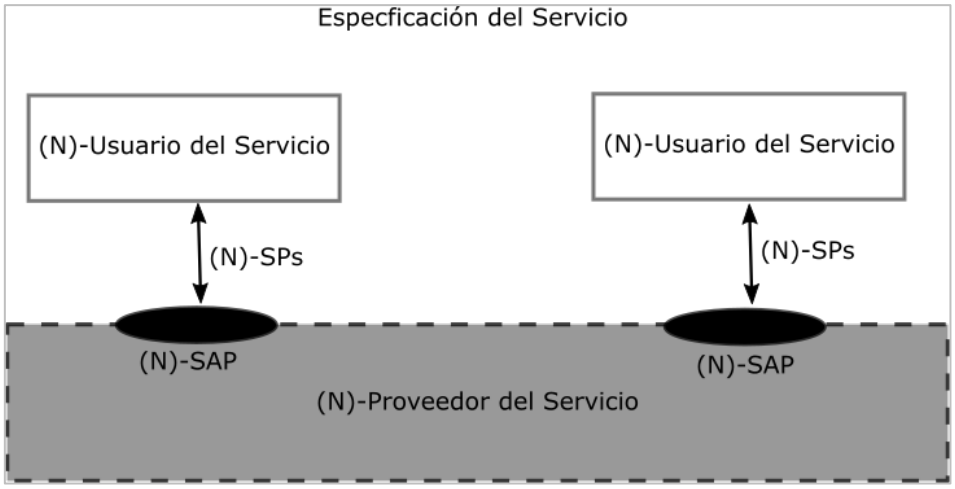
2.1.1.2.4.1.1. Especificación del Servicio

La especificación del servicio tal como la define (König, 2012), permite describir como los usuarios hacen uso del servicio y sus interfaces. Solo considera las interacciones entre usuarios del servicio, proveedores del servicio e interfaces. Esta especificación contiene la descripción parcial o total de los servicios brindados más no su comportamiento.

Para (Lai & Jirachiefpattana, 1998), cada capa de red, es como una caja negra que provee un conjunto de servicios a sus usuarios para poder interactuar con otros usuarios. A estos usuarios les preocupa la naturaleza

del servicio entregado y no les preocupa como el protocolo lo hace. En este contexto, la especificación del servicio es la descripción del comportamiento de las entradas y salidas de la capa de protocolos.

Gráfico 3: Procedimiento de Especificación del Servicio



Fuente: (Lai & Jirachiefpattana, 1998, pág. 15)

Esta especificación de servicio contiene según (König, 2012), la descripción de los servicios brindados junto a sus primitivas de servicio (SP) y sus parámetros así como toda la dependencia entre ellos. El usuario del servicio invoca al servicio mediante puntos de acceso denominados SAP (Service Access Point) mientras que el transporte de los datos lo realiza el proveedor del servicio.

Así mismo, (Lai & Jirachiefpattana, 1998) indican que una especificación de servicio está basado en primitivas de servicio denominadas SP (Service Primitives), que describen las operaciones con el proveedor de servicio y que estas operaciones no pueden ser en orden aleatorio, sino que deben estar en un orden pre-establecido.

La invocación de un servicio, en palabras de (König, 2012) es descrito a través de las primitivas de servicio (SP) y que en el contexto del modelo de referencia OSI, se desarrolló una notación sencilla para especificarlas mediante: un *nombre*, un *tipo* y sus *parámetros*. El *nombre* de una primitiva de servicio debe reflejar el propósito del servicio. Por ejemplo.

CONNECT (configurar una conexión)
DISCONNECT (liberar una conexión previamente establecida)
DATA (transmisión de datos)
ABORT (Abortar una conexión)

Por otro lado, el *tipo* especifica la función de la primitiva, de los cuales se distinguen cuatro

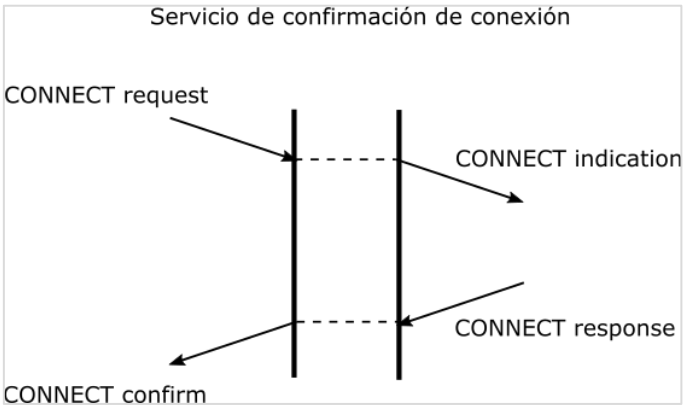
request (invocación del servicio en un SAP)
indication (relativo a la indicación en el punto de acceso par)
response (respuesta a una primitiva de tipo indication)
confirm (confirmación de la invocación del servicio a un request)

Las primitivas de servicio del tipo request y response son generadas solamente por el usuario mientras que las del tipo indication y confirm son respuestas del proveedor del servicio. Los *parámetros* son utilizados para enviar datos entre el usuario del servicio y el proveedor. Los parámetros más comunes son direcciones de red, opciones o datos del usuario. Como ejemplo, podemos observar una primitiva de servicio según la notación mencionada.

CONNECT request(source address, source dest, QoS options)
DATA indication(user info)
DISCONNECT request(session id)

Finalmente, la técnica de especificación utilizada en los estándares OSI para describir las dependencias entre primitivas de servicio son los diagramas TS (time-sequence), en donde la secuencia de interacciones se especifica a lo largo de una linea de tiempo dirigida hacia abajo.

Gráfico 4: Diagramas TS (time sequence)



Fuente: (König, 2012, pág. 9)

Los diagramas TS en algunos casos, según (König, 2012), son insuficientes para representar todas las dependencias enter las primitivas de servicio y los puntos de acceso; otras formas de descripción pueden ser consideradas como los diagramas de estado y las matrices o tablas de representación.

Por ello, (König, 2012) propone un lenguaje de modelado denominado **Level S** para abordar la descripción formal de los diferentes elementos de la especificación del servicio cubriendo tanto su sintaxis como su semántica. Este lenguaje combina los conceptos del servicio junto con algunas construcciones de lenguajes de programación de alto nivel, y permite hasta 3 niveles de descripción relacionados (servicio, protocolo y mapeo de protocolo en capas) para especificar servicios, primitivas de servicio (SP), puntos de acceso al servicio (SAP) y comportamiento del servicio via eventos internos. La sintaxis de Level S se detalla en la siguiente plantilla:

specification <nombre del protocolo>{
service <nombre del servicio>:
requested SP(<parametro: tipo de dato [**optional**], …>)
responded SP(<parametro: tipo de dato [**optional**], …>)

sap <nombre del punto de acceso>{
signal <nombre de la señal, …>
phase <nombre de la fase>:<nombre SP,…>;

```
            par event{
                (par event 1 [and condition]: par action 1 || …
                [wait event{
                    event 1 [and condition]: action 1 | …
                    event n [and condition]: action n
                }] ||
                par event n [and condition]: par action n |
                cause | triggering event)
            }
        }
}
```

Tabla 1

Sentencias del lenguaje Level S

Sentencia	Descripción Formal
spectification	Permite especificar el nombre del protocolo.
service	Permite especificar el servicio entregado.
requested	Primitiva de servicio invocado por el usuario del servicio.
responded	Primitiva de servicio desencadenado por el proveedor del servicio.
optional	Marcador de parámetro opcional.
sap	Permite especificar el nombre del punto de acceso y definir las interacciones con su sap par.
signal	Permite especificar un conjunto de eventos internos para describir diferentes estados del proveedor del servicio.
phase	Permite especificar un conjunto de primitivas de servicio que ocurren en una fase. Una fase es un subconjunto de procedimientos del protocolo que pueden ser activados mediante la sentencia set.
par event	Permite especificar una sección de eventos que pueden ocurrir de forma simultanea y describir el comportamiento de un punto de acceso ante esos eventos. El simbolo "||" es usado para denotar una ejecución concurrente. Esta sentencia se ejecuta en un bucle infinito.
wait event	Permite especificar diferentes reacciones no concurrentes a un mismo evento. La sentencia wait event puede ser

	anidada o estar dentro de una sentencia par event. El simbolo "|" es usado para denotar un comportamiento alternativo. La sentencia wait event solo reacciona al primer evento que ocurre y luego sale de su bucle.
set	Permite especificar la activación o cambio de una fase del protocolo.
and *condition*	Permite especificar las condiciones asociadas a un evento para que se pueda ejecutar sus acciones. La condición puede ser el valor de algunos parámetros o la fase actual del protocolo.
respond	Permite especificar el envio de una primitiva de servicio desde el proveedor del servicio hasta el usuario de destino. La sentencia respond es desencadenada por un evento interno. Se utiliza como parte de una acción.
event	Permite especificar un evento. Los eventos pueden ser primitivas de servicio o eventos internos del proveedor de servicio.
action	Permite especificar un conjunto de sentencias que deben ser ejecutadas en forma secuencial en respuesta a un evento. Las sentencias pueden ser: wait event, set, respond o un *PartnerSAP event*. Las acciones son separadas entre si por una nueva linea.
cause	Permite especificar la dependencia entre eventos locales y los puntos de acceso de servicio foraneo. Su sintaxis es la siguiente: *local event*: ↪*PartnerSAP.event*. La sentencia cause suele estar dentro de un par event.
triggering event	Permite especificar la contraparte de la sentencia *cause*. Es decir, para un evento generado desde el emisor utilizando la sentencia cause, existe un triggering event en el receptor. La sintaxis es: ↪*PartnerSAP.event: action.* La sentencia triggering event suele estar dentro de un par event.
PartnerSAP event	Permite especificar un evento en el punto de acceso del servicio (SAP) del otro extremo (Partner). El evento por lo tanto no ocurre en la entidad local. El PartnerSAP event utiliza el simbolo "↪" delante del nombre del evento para

evitar confundirlo con un evento local.

Fuente: (König, 2012, págs. 13-21)

Al final, (König, 2012) explica que las técnicas de descripcion formal (FDT) demandan una definición formal de la sintaxis y la semántica que cumplan con importantes requisitos: Una gran expresividad para poder representar todos los elementos del servicio y el protocolo, un razonable nivel de abstracción que evite referencias a posibles implementaciones, y la presencia de características adecuadas de lenguaje y estructuras para poder plasmar con precisión el diseño de un protocolo. Su trabajo realizado con Level S cumple estos criterios y son una respuesta a otras FDT antiguas que han intentado estandarizar esta fase del diseño con mayor o menor exito. Entre ellas tenemos a Estelle, LOTOS y SDL.

2.1.1.2.4.1.2. Especificación del Protocolo

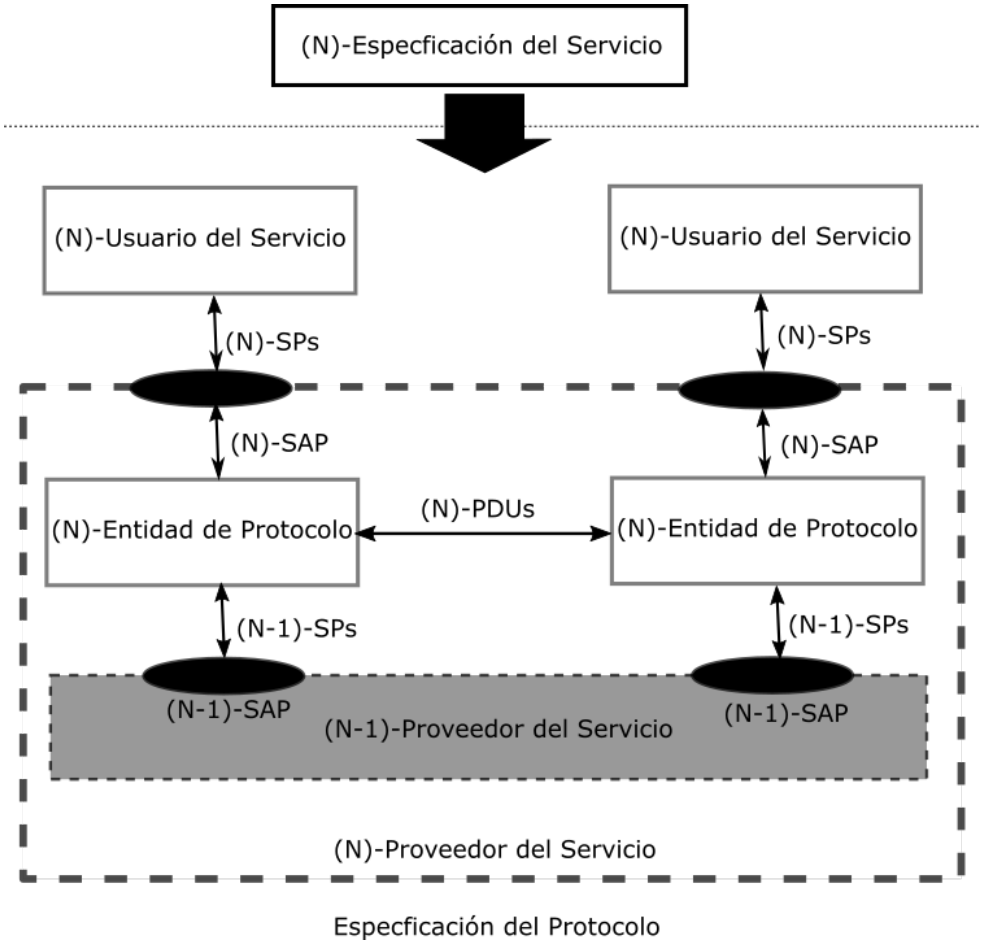
Un protocolo, según (Lai & Jirachiefpattana, 1998), consiste en un número de mensajes que son pasados entre entidades pares que establecen reglas de diálogo para el intercambio de información. La especificación del protocolo debe describir las operaciones de cada entidad en respuesta a las órdenes de los usuarios, de mensajes de otras entidades mediante unidades de datos o PDU (Protocol Data Units), intercambio de interacciones vía primitivas de servicio (SP) o vía eventos internos.

En su investigación, (König, 2012) menciona que la especificación del protocolo prescribe como debe ser brindado el servicio por las entidades pares. Indica el orden de las interacciones entre los pares así como el formato y las unidades de datos intercambiados. Consiste en una implementación abstracta de la especificación del servicio.

Es importante recalcar que (Lai & Jirachiefpattana, 1998) explicaban que el proceso que sigue la especificación del protocolo comienza con el input generado por la especificación del servicio de la capa N, es decir, se utiliza el resultado diseñado en la fase anterior para abordar detalles más técnicos pero aun lo suficientemente abstractos para no ser asociado con

ninguna implementación final. Este proceso describe que servicios entregan las entidades del (N)-protocolo a los usuarios del servicio mediante los puntos de acceso al servicio (SAP) que son observadas por las primitivas del servicio (SP) en un orden determinado. La especificación del protocolo describe un conjunto de entidades de protocolo que pueden obtenerse al refinar la especificación del servicio. Esas entidades se comunican con sus pares mediante el proveedor del servicio de la capa inferior (N-1) donde se realiza el proceso similar para dicha capa, y así sucesivamente hasta llegar a la capa final o física. El intercambio de información entre entidades de protocolo se da mediante PDUs de su respectiva capa. En consecuencia, la especificación del protocolo viene a ser la implementación de la especificación del servicio.

Gráfico 5: Procedimiento de Especificación de Protocolo



Fuente: (Lai & Jirachiefpattana, 1998, pág. 15)

Así pues, (König, 2012) refiere que las entidades de protocolo son objetos activos del proveedor de servicio que utilizan mensajes para comunicarse con su entorno. Las entidades reciben y analizan las primitivas de servicio (SP) que han sido gestionadas mediante los puntos de acceso al servicio (SAP) y mediante la información de direccionamiento de la SP, se comunica con la entidad par al otro extremo que provee los SAPs relacionados a su servicio, es decir con su entidad par.

Estos mensajes denominados PDUs (Protocol Data Units), a criterio de (König, 2012), deben ser definidos por el mismo protocolo y conocido por ambas entidades para garantizar una correcta interpretación. Los datos que lleva el PDU son los datos del usuario emitidos por las primitivas de servicio (SP); estos datos no son accedidos ni modificados por el proveedor de servicio para garantizar el principio de transparencia, es decir que los datos recibidos por la entidad par no han sido alterados. Un PDU a su vez está formado por el SDU y el PCI; donde el SDU (Service Data User) es la información del usuario y el PCI(Protocol Control Information) que son campos de control que acompañan al SDU como un mecanismo para implementar el principio de transparencia. Algunos campos de control típicos son la dirección de origen y destino, la longitud del PDU, parámetros de calidad (QoS), etc. Estos campos de control son retirados por la entidad par cuando recibe el PDU y antes de que el SDU sea entregado al usuario del servicio.

Al igual que con la especificación del servicio, (König, 2012), propone un lenguaje de modelado denominado **Level P** para describir de manera formal las interacciones entre las entidades de un protocolo asumiendo un canal virtual de comunicación abstracto entre ellas. Esta FDT abarca los conceptos de PDU y los procedimientos del protocolo con una sinxtaxis y semántica propia que se describe a continuación.

specification <nombre del protocolo>{
service <nombre del servicio>:
message{
    <PDU 1>: **struct**(
        <componente 1>:<**datatype**> [**optional**], …

<componente n>:<**datatype**> [**optional**], …

)

<PDU n>: **…**

}

protocol

<sub-protocol-name 1>:<**protocol-part**> ⇔ <**protocol-part**> [⇔ …]

<sub-protocol-name 2>:<**protocol-part**> ⇔ <**protocol-part**> [⇔ …]

entity <nombre de la entidad>

sap <nombre del punto de acceso>

signal <nombre de la señal, …>

var <nombre de la variable>:<**datatype** [**init**(value)]>,…;

timer <nombre del timer>:<**time interval**>,…

par event{

trigger event 1 [**and** [**not**] condition]**:** < protocol part> || …

trigger event n [**and** [**not**] condition]**:** < protocol part>

} // end - par event

protocol part <nombre del protocol part> (**peer** <nombre de entidad>)

[pp-signal <nombre de la señal, …>**]**

[var <nombre de la variable>:<**datatype** [**init**(value)]>,…;**]**

[timer <nombre del timer>:<**time interval**>,…]

<evento>: **begin**

<action part>

end

} // end-specification

Tabla 2

Sentencias del lenguaje Level P

Sentencia	**Descripción Formal**
spectification	Permite especificar el nombre del protocolo.
service	Permite especificar el servicio entregado.
message	Permite especificar los PDU que utiliza el protocolo
struct	Permite especificar la estructura interna del PDU. Para acceder a sus componentes, se utiliza el nombre del PDU

seguido de un punto (.) y el nombre del elemento.

datatype Permite especificar el tipo de dato de un componente o elemento del PDU. Los tipos de datos soportados son similares a los de un lenguaje de programación: integer, boolean, address, bits, byte, range(x..y), array [n] of, etc.

optional Marcador de parámetro opcional.

protocol Permite especificar que protocol parts forman un protocolo. El protocolo puede estar formado por sub-protocolos. Por cada sub-protocolos se listan los diferentes protocol-parts que la componen, separados por el carácter ⇔

protocol-part Permite especificar el nombre de protocol part asociado a un sub-protocolo. Un protocol-part describe el comportamiento en la comunicación de una entidad con su par, es decir los protocol-parts de ambas entidades constituye el protocolo en si. El protocol-part se compone de nombre de la entidad y el nombre del part.

entity Permite especificar el nombre de la entidad y describir el comportamiento de las entidades. Se compone de la especificación sap y las declaraciones signal, var, timer y par-event

sap Permite especificar el nombre del punto de acceso que es entregado por la entidad. Este punto de acceso debe estar relacionado con el SAP declarado en la especificación Level S..

signal Permite especificar la ocurrencia de un conjunto de eventos internos en los protocol parts.

var Permite especificar variables al estilo de un lenguaje de programación. Las variables deben estar inicializadas mediante la sentencia init.

init Permite inicializar las variables definidas con la sentencia var. Su sintaxis es init(*value*). Donde *value* corresponde al valor de inicialización de una variable.

timer Permite especificar temporizadores utilizados por la entidad para el monitoreo o generación de algun evento particular.

time interval	Permite definir el intervalo de tiempo de un timer. Las unidades de tiempo pueden ser milisegundos (ms), segundos (s), minutos (m) u horas (h). La sintaxis del time interval es: *init_time*..(?|*end_time*), donde min_time y max_time corresponden al tiempo de inicio y fin del intervalo. El carácter ? es empleado cuando se desconoce el fin del intervalo del timer.
par event	Permite especificar la ejecución paralela de los protocol parts. El simbolo "||" es usado para denotar una ejecución concurrente. Esta sentencia se ejecuta en un bucle infinito. La ejecución de un protocol part se consigue mediante la evaluación de un trigger event que puede incluir alguna condición para su ejecución
trigger event	Permite especificar el nombre de un evento proveniente de una primitiva de servicio (SP), PDU, timeout, signal y otros.
and [**not**] condition	Permite especificar las condiciones asociadas a un evento para que se pueda ejecutar sus acciones. La condición puede ser la condicional y asi como su negación.
protocol part	Permite especificar el comportamiento de un protocol part. A diferencia del protocol-part utilizado en la sección protocol, la sentencia protocol part no solo define un nombre sino tambien su comportamiento. Al igual que la sentencia entity, se compone de las declaraciones signal, var, timer y además incluye un *action part*.
peer	Permite especificar el nombre de la entidad par del protocol part
pp-signal	Permite especificar todos los signal que se esperan en su protocol part
action part	Permite especificar las acciones que se llevaran a cabo en el protocol part. El action part consiste en trigger events y de sus reacciones a esos eventos. Estos eventos deben ser los mismos que fueron definidos en la sección entity. La reacción esta incluida dentro de las cláusulas begin y end y consiste en un conjunto de sentencias que son ejecutadas de forma secuencial. Las sentencias

pueden ser: send, par event, wait event, respond, set, time-out, start, reset, if, case, loop, skip, exit, exit when, incr, decr, cancel protocol y coding.

send Permite especificar la interacción con la entidad par mediante el envio de mensajes. El mensaje debe estar especificado en la sección message. El nombre de la entidad receptora debe corresponder con el nombre del protocol part. El mensaje tiene la sintaxis para la entidad que envia el mensaje: <PDU> → <protocol part partner>. El mensaje tiene la sintaxis para la entidad que recibe el mensaje: <PDU> ← <protocol part partner>.

wait event Permite especificar diferentes reacciones no concurrentes a un mismo evento. La sentencia wait event puede ser anidada o estar dentro de una sentencia par event. El simbolo “|” es usado para denotar un comportamiento alternativo. La sentencia wait tiene lugar en el protocol part de la entidad. Los Triggering events pueden ser: SP, PDU, time-outs y signals y pueden tener condiciones.

respond Permite especificar el envio de una primitiva de servicio desde el proveedor del servicio hasta el usuario de destino. La sentencia respond es desencadenada por un evento interno. Se utiliza como parte de una acción.

set Permite especificar el envio de una señal a otro protocol part. La señal una vez atendida, vuelve a su valor original (false) hasta que vuelva a ser activada.

time-out Permite especificar un evento que se desencadena si otro evento no ocurre en el tiempo esperado. Es un mecanismo de reacción alternativo para una acción esperada para prevenir un deadlock.

start Permite especificar el inicio de un timer. Los timers declarados en una entidad son validos para todos los protocol parts, pero un timer declarado en un protocol part solo tiene validez de forma local a ese protocol part. Si se utiliza la sentencia start en un timer que aun no termina, el timer se resetea y reinicia.

reset Permite especificar la parada de un timer. Si el timer esta

	detenido, no tiene ningun efecto.
if	Permite especificar la evaluación de una condición
case	Permite especificar la evaluación de multiples condiciones
loop	Permite especificar un bucle
skip	Permite especificar una sentencia vacia
exit	Permite especificar la salida de un bucle o un wait event
exit when	Permite especificar una condición de salida para la sentencia exit
incr	Permite especificar el incremento de un valor
decr	Permite especificar el decremento de un valor
cancel protocol	Permite especificar la cancelación de la ejecución del protocolo. Termina todos los protocol parts, timers y eventos.
coding	Permite especificar la codificación y decodificación de los PDU. La sintaxis es la siguiente: coding_*PDU*(*datos,…*) donde PDU es el nombre del PDU a codificar y datos son las variables que serán enviadas.

Fuente: (König, 2012, págs. 34-43)

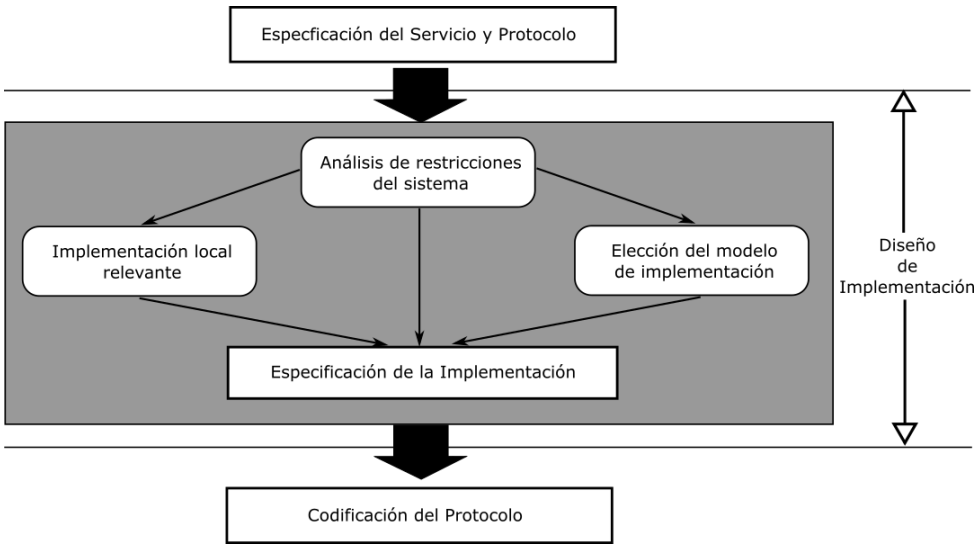
Para (König, 2012), el desarrollo de las FDT condujo el desarrollo de técnicas como complemento a las FDT, entre ellas la más importante fue la notación de sintaxis abstracta ASN.1 que fue diseñada para resolver el problema de la representación de datos variante entre diferentes sistemas de computadoras debido a su arquitectura o a los lenguajes utilizados. ASN.1 permitió el intercambio heterogéneo de los datos y es una de las formas más utilizadas para enviar los PDU con la garantía de ser recibidos de forma correcta.

2.1.1.2.4.1.3. Especificación de Implementación

En su investigación, (König, 2012) propone una tercera fase en el diseño de protocolos, que consiste en una implementación en forma de un prototipo. No se trata de una implementación completa que se lleva a cabo en la fase de codificación, sino más bien de una implementación en un entorno de ejecución concreto con el objetivo de validar los principales procedimientos del protocolo durante su diseño. A diferencia de las anteriores fases, no se contempla una técnica de descripción formal; el resultado de esta fase es precisamente el mismo prototipo.

El mayor problema en la implementación es el mapeo de la funcionalidad del protocolo hacia un sistema objetivo. La calidad y eficiencia de un protocolo está determinada casi siempre por la implementación antes que por el diseño. Esto se debe a que las especificaciones del protocolo son abstractas y brinda la total libertad a los implementadores para un diseño individual. Así mismo, las implementaciones del protocolo están determinadas por el entorno de ejecución y no existe una receta universal para una implementación ventajosa de cualquier protocolo. Sin embargo, existe un conjunto de directivas y técnicas que pueden ser utilizadas para tomar mejores decisiones en esta fase (König, 2012).

Gráfico 6: Pasos de la Especificación de la Implementación



Fuente: (König, 2012, pág. 352)

“El objetivo del diseño de implementación es mapear de forma óptima el diseño de protocolo, documentado en las especificaciones de servicio y protocolo hacia el entorno de ejecución de un sistema objetivo”. (König, 2012, p. 352).

Es en esta fase, según (König, 2012), que es necesaria porque el modelo lógico de la especificación del protocolo no siempre se preserva en la implementación debido a que las sus estructuras pueden ser ineficientes y también porque las especificaciones del protocolo pueden contener opciones y lineamientos que deben ser decididos exclusivamente en un nivel de implementación.

El análisis de restricciones del sistema consiste en una evaluación directa de las características y restricciones del entorno de ejecución en donde se implementará el protocolo. Esta evaluación contempla el sistema operativo a usar, los posibles lenguajes de programación y las librerías existentes (König, 2012).

La implementación local relevante se refiere a las decisiones de diseño que en la especificación del protocolo fueron dejadas a manos del implementador. Estas decisiones están relacionadas usualmente con las interfaces al entorno de ejecución, opciones del protocolo y elecciones de implementación. El implementador debe realizar decisiones concretas para estas posibles opciones (König, 2012).

La elección del modelo de implementación determina como la implementación es mapeada en la estructura del proceso del entorno de ejecución. Este mapeo no necesariamente es de uno a uno (por ejemplo un proceso o hilo por entidad), puede ser de uno a muchos (por ejemplo varias entidades por proceso). Existen 2 modelos de implementación básicas: el modelo servidor y el modelo de actividad por hilos. El modelo servidor es el más conocido y permite tener un proceso que espera en un bucle a los eventos, y cuando ocurren entonces el proceso los analiza y emite una respuesta. El modelo de actividad por hilos permite implementar las entidades mediante un conjunto de procedimientos que se ejecutan en paralelo. Cada procedimiento procesa un evento. Esta concurrencia sin

embargo requiere de un cuidado especial con la sincronización (König, 2012).

Finalmente, la especificación de la implementación forma las bases para la siguiente fase del proceso de desarrollo de protocolos que es la codificación. A diferencia de la especificación del protocolo, esta especificación está relacionada con el sistema objetivo. Esta especificación no está asociado a ninguna FDT sino que es trabajada como un documento interno a criterio del implementador (König, 2012).

2.1.1.2.4.2. Metodología de diseño heurística

"Hoy en día el desarrollo empírico de protocolos sigue prevaleciendo, utilizando métodos de diseño heurístico en mayor o menor grado. En su mayoría en universidades o institutos de investigación mediante documentos estandarizados". (König, 2012, p. 289).

"El desarrollo de nuevos protocolos ha probado ser más flexible porque no han sido desarrollados en organismos de normalización, sino en grupos de trabajos de expertos. Las propuestas son publicadas en las RFC y están abiertas a discusión mundial". (König, 2012, p. 66).

La principal diferencia con el diseño formal es la combinación de las especificaciones de servicio y protocolo. En la metodología formal, estas descripciones son separadas de forma estricta pero esto no sucede con los RFCs (*Request for Comments*) de la IETF en donde las especificaciones de servicio no se describen de forma explícita y junto con las especificaciones del protocolo, son descritas de manera informal mediante el uso de descripciones textuales de procedimientos del protocolo (König, 2012).

2.1.1.2.4.2.1. Request for Comments (RFC)

"Los estándares juegan un papel importante en el proceso de desarrollo de protocolos. Todos los protocolos usados hoy en redes abiertas son

publicados como estándares. Los protocolos de internet están publicados en RFCs. La mayoría descrita de manera informal". (König, 2012, p. 287).

"Un RFC es un reporte simple, originalmente llamado *Request for Comments* debido a que los investigadores publicaban sus propios resultados, teorías y actividades, y solicitaban respuestas de otros investigadores a través de este mecanismo". (Loshin, 1999, p. 14).

Los RFC pueden ser escritos por cualquiera, ya sean estudiantes, profesores, investigadores o empleados del sector de networking. Mientras el documento sea relevante para las comunicaciones, este en el formato correcto y enviado según la normativa establecida, entonces tiene una oportunidad de ser publicado como un RFC (Loshin, 1999).

Para que un RFC se convierta en STD se sigue un procedimiento de aprobación similar a una ISO: "una propuesta inicial se examina como *Proposed Standard*. Después de un mínimo de seis meses, podrá ser aceptado como *Draft Standard*. Por último, después de al menos otros cuatro meses, puede aceptarse como *Standard Protocol* para uso de Internet". (Sharp, 2008, p. 374)

Los RFC tienen 6 status o niveles de madurez según (Loshin, 1999). Un *Standard Protocol* o *Internet Standard* corresponde a un protocolo que ha sido categorizado como un estándar de Internet oficial y establece como se deben realizar las cosas que aborda el protocolo. Un *Draft Standard* corresponde a un protocolo que se encuentra en permanente observación y posibles modificaciones para poder convertirse en estándar. Se requiere que existan varias implementaciones para poderlo evaluar. Este nivel de madurez fue retirado en el RFC 6410. Un *Proposed Standard* es un protocolo propuesto para ser considerado un futuro estándar. Deben ser implementados y probados como parte de su revisión. Un *Informational Protocol* es un protocolo desarrollado fuera de la comunidad de Internet, usualmente por empresas privadas que son publicadas como RFC. Un *Experimental Protocol* es un protocolo que ha sido considerado para experimentos y no deben ser implementados fuera del grupo a cargo del

protocolo. Un *Historical Protocol* es un protocolo que ya no es relevante o que ha sido sobrepasado por otros protocolos.

Así mismo, “algunos RFCs pueden ser categorizados como *Best Current Practice* en algún área; también se les da un número adicional, que en este caso identifica el documento en la serie de mejores prácticas actuales (BCP). Por ejemplo, RFC2026 es BCP9” (Sharp, 2008, p. 374).

Cada RFC tiene un número asociado que le fue asignado según el orden en que apareció. Es importante anotar que un STD no es un documento diferente, sino que es un número adicional al otorgado previamente a uno o varios RFC. Por ejemplo el RFC 793 (Protocolo TCP) es también el estándar de Internet STD 7. (Sharp, 2008).

Tabla 3

Muestra de documentos RFC

Número	**Titulo**	**Status**
RFC 791 alias STD 5	Internet Protocol (IP)	Internet Standard
RFC 793 alias STD 7	Transmission Control Protocol (TCP)	Internet Standard
RFC 792 parte del STD 5	Internet Control Message Protocol (ICMP)	Internet Standard
RFC 959 alias STD 9	File Transfer Protocol (FTP)	Internet Standard
RFC 821 parte de STD 10	Simple Mail Transfer Protocol (SMTP)	Internet Standard
RFC 5234 alias STD 68	Augmented BNF for Syntax Specifications (ABNF)	Internet Standard
RFC 5681	TCP Congestion Control	Draft Standard
RFC 5322	Internet Message Format	Draft Standard
RFC 3912	WHOIS Protocol Specification	Draft Standard
RFC 2865	Remote Authentication Dial In User Service (RADIUS)	Draft Standard
RFC 1541	Dynamic Host Configuration Protocol (DHCP)	Proposed Standard
RFC 7617	The 'Basic' HTTP	Proposed Standard

	Authentication Scheme	
RFC 7072	A Reputation Query Protocol	Proposed Standard
RFC 3384	Lightweight Directory Access Protocol (version 3) Replication Requirements	Informational
RFC 3717	IP over Optical Networks: A Framework	Informational
RFC 7322	RFC Style Guide	Informational
RFC 6928	Increasing TCP's Initial Window	Experimental
RFC 3700	Internet Official Protocol Standards	Historic (cambiado de Internet Standard en Mayo 2008)
RFC 4107 alias BCP 107	Guidelines for Cryptographic Key Management	Best Current Practice
RFC 2026 parte de BCP 9	The Internet Standards Process -- Revision 3	Best Current Practice

Fuente: (IETF,IRTF,IAB)

2.1.1.2.4.2.2. Guía de estilo RFC

Como parte de los documentos RFC, existe una guía de estilo para elaborarlos el cual esta descrita en el RFC 7322 (RFC Style Guide). “Este documento describe las convenciones de estilo fundamentales y únicas y las políticas editoriales actualmente en uso para la serie RFC. Captura los requisitos básicos del editor RFC y ofrece orientación sobre el estilo y estructura de un RFC” (Flanagan & Ginoza, p. 1).

“El objetivo final del proceso de publicación de RFCs es producir documentos legibles, claros, consistentes y razonablemente uniformes. Las convenciones básicas de formato fueron establecidas en la década de 1970 por el editor original de los RFC, Jon Postel” (Flanagan & Ginoza, p. 3).

“El editor del RFC es responsable de la gestión editorial y publicación de la serie RFC” (Loshin, 1999, p. 29). Su trabajo es seguir en su mayoría las reglas generalmente aceptadas en el mundo de las publicaciones técnicas como el CMOS (Chicago Manual of Style); dichas reglas consideran aspectos como la gramática, puntuación, capitalización, longitud de parrafos, citas, etc. Asi mismo, tambien aplica ciertas excepciones para garantizar la claridad técnica las cuales forman parte del documento que lo guia: el RFC 7322. (Flanagan & Ginoza, p. 3).

(Flanagan & Ginoza) indican que entre las convenciones de estilo mas importantes, destaca el idioma a utilizar que debe ser exclusivamente el idioma ingles. En cuanto a la estructura, un documento RFC esta compuesto por los siguientes elementos:

Tabla 4

Estructura RFC

Sección	**Requerido**	**Aporte RFC Editor**
First-page header	Sí	Sí
a. Author/Editor	Sí	
b. Organization	Sí	
c. "ISSN: 2070-1721"		Sí
d. Updates and Obsoletes		Sí
Title	Sí	
Abstract	Sí	
RFC Editor Note	Sí	Sí
Status of This Memo	Sí	Sí
Copyright Notice	Sí	Sí
Table of Contents	Sí	Sí
Body of the Memo	Sí	
a. Introduction	Sí	
b. Requirements Language (RFC 2119)		
c. MAIN BODY OF THE TEXT	Sí	
d. IANA Considerations		
e. Internationalization Considerations		
f. Security Considerations	Sí	
g. References		

g.1. Normative References	
g.2. Informative References	
Appendix A.	
Appendix B.	
Acknowledgements	
Contributors	
Author’s Address	Sí

Fuente: (Flanagan & Ginoza)

2.1.1.2.4.2.3. Guía RFC sobre diseño de protocolos

El RFC 3117 titulado “On the Design of Application Protocols” cubre algunos aspectos relacionados con el diseño técnico de protocolos al describir uno nuevo denominado BXXP (Blocks eXtensible eXchange Protocol) ante la necesidad de tener un framework de protocolo de aplicación genérico.

Para (Rose, 2001), el protocolo SMTP sería la guia casi perfecta sobre lo que debe ser un protocolo de aplicación y su diseño, salvo por el hecho de haber sido creado hace mas de 30 años y que desde entonces no han habido mejoras significativas en el reuso del comportamiento funcional de un protocolo.

Asi mismo, (Rose, 2001) brinda algunas pautas relacionadas con el diseño desde el mismo momento de su concepción. Y establece que para proceder a diseñar un protocolo se debe tener en cuenta el siguiente proceso:

a. Buscar un protocolo de intercambio existente que haga mas o menos lo que uno quiera.
b. Definir un modelo de intercambio en la parte superior de la infraestructura web que haga mas o menos lo que uno quiera.
c. Definir un modelo de intercambio en la parte superior de la infraestructura de correo electrónico que haga mas o menos lo que uno quiera.
d. Definir un nuevo protocolo desde cero que haga exactamente lo que uno quiera.

Los protocolos, según la explicación de (Rose, 2001), pueden diseñarse en base a la simplificación de sus problemas y que el paso mas importante es establecer límites a esos problemas mediante la asignación de las siguientes características al protocolo: Orientación a la conexión, peticiones y respuestas para el intercambio de mensajes y el soporte de mensajes asincronos.

Una vez limitado el problema, se deben diseñar los mecanismos del protocolo y validar algunas propiedades de diseño. Los mecanismos del protocolo son las tareas que una aplicación del protocolo debe efectuar y ver como afectan en su rendimiento. Las areas de interes mas destacadas son: *framing* (como empieza y termina un mensaje), *encoding* (como se representa un mensaje cuando es intercambiado), *reporting* (como se describen los errores), *asynchrony* (cuan independiente se tratan los mensajes), *authentication* (como se identifican y verifican los pares) y *privacy* (como se protegen los mensajes intercambiados de interferencia o modificación). Finalmente se revisan las principales propiedades del protocolo que son: escalabilidad, eficiencia, simplicidad, extensibilidad y robustez. (Rose, 2001).

Los documentos RFC a diferencia de las metodologías formales, estan centrados en aspectos mas técnicos, mezclando tanto las especificaciones del servicio como del protocolo. Los documentos RFC son guias prácticas importantes que son resultado de la experiencia directa con el diseño de protocolos, por lo que resulta ser de gran ayuda tenerlas en cuenta a la hora de la fase de diseño.

2.1.1.3. Dimensión metodológica

El autor ha creído conveniente establecer una guía metodológica propia que sirva de base para la operacionalización de la variable independiente, debido a que la variable a tratar corresponde con un tópico poco estudiado en el mundo académico y además porque la medición a través de sus indicadores puede resultar confusa debido a la naturaleza técnica de la materia estudiada.

El autor presenta un enfoque de 2 dimensiones metodológicas denominadas *selección* y *diseño*. Este enfoque es una propuesta al trabajo de investigación relacionado con el diseño de nuevos protocolos para cualquier capa de red y

que ha servido para delimitar apropiadamente el problema a abordar y configurar los indicadores apropiados para su medición.

2.1.1.3.1. Selección

Es una dimensión metodológica que consiste en delimitar el alcance que tiene el protocolo dentro de su entorno tecnológico básico. Este entorno tecnológico básico reside en dos aspectos: la población de servidores y los sistemas y servicios que ofrece esta población. Con esta información se puede elaborar una muestra de trabajo que sirve de punto de partida para abordar el diseño de un protocolo informático.

Obtener la población de servidores donde se ejecutan los sistemas y servicios debe ser el primer paso para delimitar el protocolo, debido a que permite tener claro un detalle importante del protocolo en su diseño: la plataforma. Diseñar un protocolo multiplataforma no es lo mismo que diseñarlo para una sola plataforma y esto se consigue de forma sencilla mediante la revisión de la población de servidores. La categoría de este indicador será **COMPLETO** si el investigador obtiene la lista oficial de servidores en la institución donde realiza la investigación; y la categoría será **IMCOMPLETO** si esta lista no es proporcionada o se encuentra incompleta.

Al identificar sistemas y servicios que se ejecutan en la población de servidores nos permite restringir aún más el ámbito de acción del protocolo. Este paso tiene un fin aún más técnico debido a que permite establecer directamente las relaciones que tendrán estos sistemas y servicios con el protocolo a diseñar. El término sistema hace referencia al tipo de sistema operativo en donde trabajará el protocolo pero no en sentido abstracto sino más bien técnico, es decir al sistema operativo en ejecución. Por otro lado, el termino servicio hace referencia a los diferentes programas que se están ejecutando en modo servidor en el sistema operativo.

Para delimitar adecuadamente estos sistemas y servicios se propone categorizarlos por la necesidad de monitoreo, es decir por la urgencia que tienen de formar parte de la solución que ofrece el protocolo. Por lo tanto, se debe conocer si se requiere o no su monitoreo. Para conseguir esta

información el autor considera de utilidad elaborar un cuestionario de identificación de sistemas y servicios con los siguientes datos:

Tabla 5

Estructura de cuestionario de identificación de sistemas y servicios

Dato	**Tipo de dato**	**Descripción**
Servidor	Cadena de caracteres	Nombre del servidor. Hace referencia al equipo físico (hardware) o virtual (virtual machine) en donde se ejecutan los sistemas o servicios.
Sistema / Servicio	Cadena de caracteres	Nombre del sistema o servicio en ejecución dentro del servidor.
Pregunta técnica	Booleano	Pregunta de carácter técnico que está muy relacionado con el propósito del protocolo y que sirve en la evaluación del monitoreo y control del sistema o servicio. Deben existir al menos 2 preguntas técnicas y no se debe utilizar la negación como parte de la pregunta.
Requiere Monitoreo y Control	Booleano	Contiene el resultado de la evaluación de todas las preguntas técnicas. Aunque la evaluación final está sujeta a la libertad del investigador, se exige que cuando todas las preguntas técnicas resulten en falso, la evaluación final también sea falsa.

Fuente: Elaboración del investigador

El tipo de dato “cadena de caracteres” del cuadro anterior, consiste en una secuencia de caracteres alfanuméricos muy similar a los de cualquier lenguaje de programación. En cuanto al tipo de dato “Booleano”, hace referencia a 2 posibles valores en cualquiera de las siguientes formas: verdadero o falso, 1 o 0, Si o No.

Finalmente, se debe elaborar la muestra de trabajo con la información recolectada previamente como paso final al proceso de selección, es decir,

después de obtener solamente aquellos servicios que requieren monitoreo y control, se procederá a evaluar su prioridad y así obtener una muestra. Esta muestra de trabajo representa un filtro más para delimitar el alcance que tiene el protocolo, y se utilizará la categorización por 3 prioridades (ALTA, MEDIA y BAJA). Al igual que en el proceso anterior, el autor ha considerado de utilidad tener en cuenta el siguiente cuestionario de priorización de sistemas y servicios.

Tabla 6

Estructura de cuestionario de priorización de sistemas y servicios

Dato	**Tipo de dato**	**Descripción**
Servidor	Cadena de caracteres	Nombre del servidor. Hace referencia al equipo físico (hardware) o virtual (virtual machine) en donde se ejecutan los sistemas o servicios.
Sistema / Servicio	Cadena de caracteres	Nombre del sistema o servicio en ejecución dentro del servidor.
Descripción	Cadena de caracteres	Breve descripción del sistema o servicio y de su función principal.
Prioridad	Opción	Contiene la prioridad de implementación del protocolo. Los valores pueden ser Bajo, Medio y Alto. Esta evaluación no tiene un carácter técnico estricto y se deja a criterio de los investigadores y participantes del proyecto la decisión final según su conveniencia.

Fuente: Elaboración del investigador

A nivel técnico estas 3 prioridades no tienen mayor relevancia, es decir forman parte de la solución del protocolo, sin embargo se trata más bien de una prioridad de carácter administrativo que nos permite concentrarnos solamente en diseñar un protocolo para la muestra con Prioridad Alta.

2.1.1.3.2. Diseño

Es una dimensión metodológica que consiste en elaborar una propuesta de protocolo mediante la aplicación de una metodología mixta que el autor ha considerado conveniente emplear, debido a que reúne aspectos teóricos y técnicos permitiendo abordar un marco de trabajo que sirve de guía para diseñar protocolos.

Esta metodología mixta consiste en unir lo más relevante de las fases de diseño propuestas por König en su metodología de diseño formal con la metodología de diseño heurística basada en las guias RFC. Esta dimensión constituye el corazón de la investigación con el objetivo de “elaborar una propuesta de protocolo”.

Para elaborar una propuesta de protocolo, es necesario cubrir las 3 fases del diseño propuestas por la metodología de diseño formal, es decir diseñar especificaciones de servicio, protocolo e implementación. Fusionando esta última con la metodología heurística debido a que son muy relacionadas mediante la elaboración de un documento técnico RFC. A estas 3 fases se le agrega una cuarta fase, que consiste en la simulación del protocolo propuesto.

La especificación del servicio debe describir la interacción del servicio, que el protocolo va a entregar, entre los usuarios del servicio, los proveedores y sus interfaces. También se debe considerar el uso de técnicas de descripción formal para representar la lógica general del protocolo. El lenguaje de modelado Level S es el recomendado por el investigador por ser el más cercano a los procedimientos de programación tradicionales. Los diagramas de estado así como los diagramas TS pueden ser considerados para un mejor entendimiento del funcionamiento a nivel de servicio del protocolo.

La especificación del protocolo debe describir la interacción de las entidades pares del protocolo, el orden de sus funciones así como el formato de sus paquetes de datos (PDU). Esta especificación es crítica porque es el puente que une las otras especificaciones (servicio e implementación). El investigador recomienda utilizar el lenguaje de modelado Level P para describir el comportamiento del protocolo.

La especificación de implementación debe describir el prototipo del protocolo en un entorno de ejecución controlado, siguiendo los 4 pasos de König: análisis de restricciones, implementación local relevante, elección de modelo de implementación y especificación de la implementación a través de la construcción de un documento RFC.

Finalmente se debe implementar el protocolo diseñado en cualquier lenguaje de programación, sin necesidad de incluir todas las características del protocolo salvo las más relevantes que puedan servir para validar la investigación deseada. El investigador recomienda utilizar el lenguaje C o el lenguaje Go por tratarse de lenguajes potentes para este tipo de tareas. Esta implementación deberá ejecutarse en un entorno controlado (por ejemplo un entorno virtualizado) y no en un entorno de producción. En este entorno se deben realizar las diferentes pruebas que sean necesarias para obtener los resultados que serán utilizados para fines de probar la hipótesis de la investigación. El ciclo de desarrollo abordado para la construcción del protocolo debe tener como mínimo el análisis, diseño, codificación y pruebas.

Si bien König considera dentro del ciclo de desarrollo de un protocolo la verificación, evaluación de performance y testing como entregables, el investigador considera no tomarlos en cuenta debido a que estas fases estan ya inmersas en el ciclo anteriormente propuesto. Considerar estas fases adicionales (como entregables independientes) puede ocasionar una complejidad excesiva e innecesaria para el diseño de protocolos simples como el abordado en esta investigación.

Para categorizar la propuesta de protocolo, se considerarán los entregables de las 4 fases de esta dimensión. Estos entregables se indican a continuación:

- **Especificación de Servicio**: diagrama TS, diagrama de estado, modelado en level S.
- **Especificación de protocolo**: modelado en level P.
- **Especificación de implementación**: análisis de restricciones, implementación local relevante, elección de modelo de implementación, RFC.

- **Simulación**: programación, despliegue de entorno controlado, pruebas.

La categorización de la propuesta de protocolo puede ser **OPTIMO** si cumple con todos los entregables de las 4 fases; será **ACEPTABLE** si solo hace falta el documento RFC; finalmente la categoría será **DEFICIENTE** si falta algún entregable de la fase de simulación.

2.1.2. Revisión de los derechos de acceso de los usuarios

2.1.2.1. Sistema Operativo Linux

Linux es un sistema operativo que convierte tu hardware de computadora en una maquina utilizable. Interactúa con todos los periféricos y efectúa tareas como almacenar archivos, mostrar información en pantalla o imprimir. Es una colección de programas que juntos hacen esto posible. El kernel de Linux es el corazón donde todas las operaciones de bajo nivel se llevan a cabo, pero por si solo no es suficiente pues requiere de esa colección de programas para funcionar (Valade, 2005).

Para conseguir todo esto, (Kalle Dalheimer & Welsh, 2006) refieren que, las distribuciones de Linux recolectan grandes cantidades de tecnología diversa, de forma especial sobre nuevos desarrollos y tecnologías de hardware. Los desarrolladores tienen acceso a todos el código que hace posible que el sistema operativo funcione, de hecho quizá el más grande proyecto de colaboración de software en la historia humana.

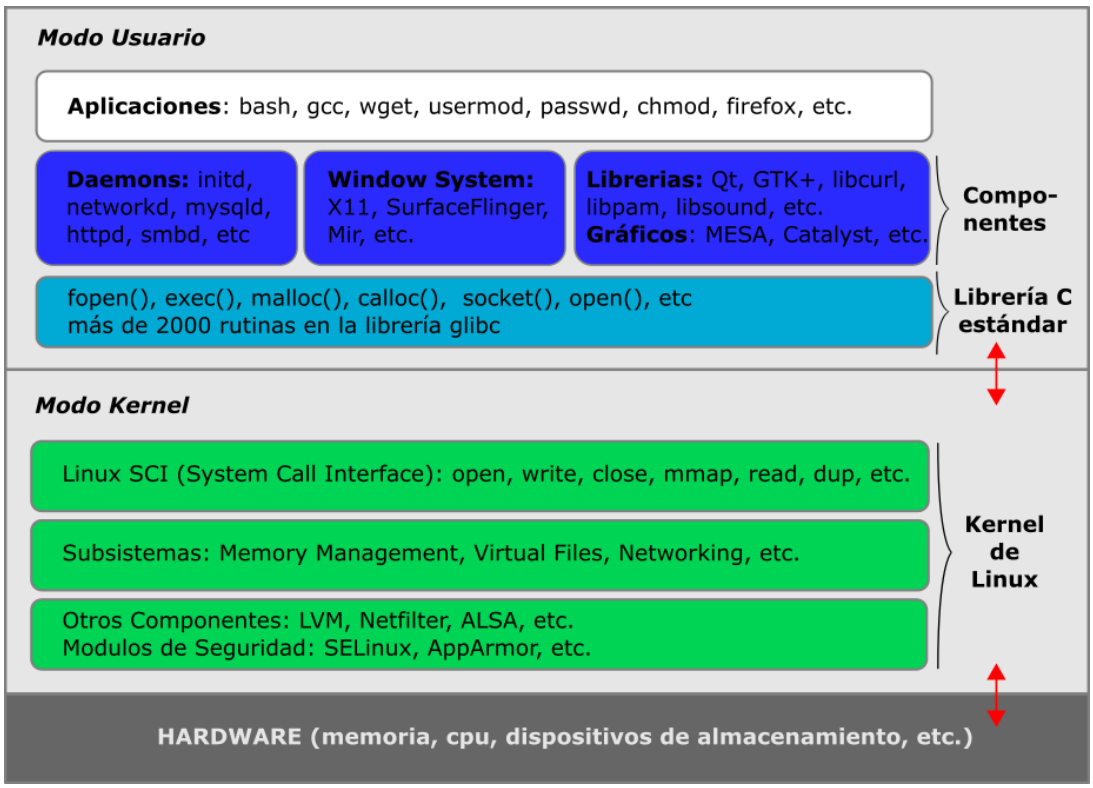
Por su parte, (Valade, 2005) refiere que Linux está basado en el sistema operativo UNIX, desarrollado en 1970, y para 1991 el estudiante Linus Torvalds de la Universidad de Helsinki decidió crear su propio sistema operativo tipo UNIX. Linus encontró en la comunidad open source la valiosa ayuda que necesitaba para sacar adelante su proyecto. (Kerrisk, 2010) sostiene que uno de los apoyos más fuertes provino del proyecto GNU liderado por Richard Stallman empezado ya en 1984 y para entonces tenía una completa suite de programas para ser utilizados o portados a sistema de tipo UNIX, lo cual popularizó enormemente al sistema Linux.

2.1.2.1.1. El kernel de Linux

El corazón de todo el sistema operativo Linux es el denominado kernel y su presencia simplifica enormemente la programación y el uso de otros programas y a su vez incrementa el poder y flexibilidad a los programadores. Para hacer esto provee una capa de software que administra los recursos limitados de una computadora (Kerrisk, 2010).

En su libro, (Love, 2010) menciona que los componentes típicos de un kernel son interruptores de servicio, un planificador de procesamiento, un sistema de gestión de memoria, servicios cono el subsistema de red o intercomunicación de procesos, etc. Debido a la naturaleza delicada del kernel, este reside en un estado de sistema elevado a comparación de las aplicaciones de usuarios, con el objetivo de proteger sus unidades de gestión de memoria. Este estado del sistema se denomina "kernel-space" y por el contrario las aplicaciones del usuario son ejecutadas en el "user-space". Ambos ven un subconjunto de recursos disponibles y pueden ejecutar ciertas funciones del sistema, acceder al hardware o a la memoria, de tal modo que cuando el kernel se ejecuta (en el kernel-space) se dice que lo hace en modo kernel, y cuando un proceso regular lo hace (en el user-space) entonces lo hace en modo usuario.

Gráfico 7: Separación entre Modo Usuario y Modo Kernel



Fuente: (Wikipedia, 2015)

2.1.2.1.2. Usuarios y Grupos del sistema Linux

Linux es un sistema operativo multiusuario, según (Valade, 2005), que permite a muchos usuarios trabajar en la misma computadora al mismo tiempo. Para ello el trabajo de cada usuario debe permanecer separado para no interferir con el trabajo de otro y prevenir accidentes o malas intenciones (borrado de información). Las cuentas Linux lo hacen posible.

Los usuarios y grupos en Linux permiten la autorización en Linux, para ello cada usuario es asociado con un entero positivo único llamado "user id" (UID). Cada proceso es asociado con un uid y así se puede identificar al usuario que ejecuta el proceso a lo cual se denomina "real uid". Dentro del kernel, el uid es el único concepto de un usuario, sin embargo los usuarios se refieren a sí mismos y a otros mediante nombres de usuarios (usernames) y no mediante números. Los usernames se almacenan en el archivo /etc/passwd y las librerías se encargan de mapearlos con sus correspondientes uids (Love, Linux System Programming, 2013).

Como complemento, (Kerrisk, 2010) refiere que cada usuario en el sistema es identificado de forma única y los usuarios pueden pertenecer a grupos mediante un “Group ID” (GID). Sus credenciales son almacenadas en forma encriptada y por razones de seguridad se almacenan en un archivo separado (/etc/shadow) que solo es leído por usuarios privilegiados. Cada usuario en el sistema es identificado de forma única y los usuarios pueden pertenecer a grupos. Sus credenciales son almacenadas en forma encriptada y por razones de seguridad se almacenan en un archivo separado (/etc/shadow) que solo es leído por usuarios privilegiados. El propósito de un id de usuario y de grupo es determinar la propiedad de varios recursos del sistema y controlar los privilegios otorgados a los procesos que acceden a esos recursos. Por ejemplo, cada archivo del sistema Linux pertenece a un usuario y grupo particular y cada proceso tienen un número id de usuario y grupo que determina quién es su propietario y que permisos tiene cuando accede a dicho archivo.

Resulta muy útil organizar usuarios en grupos para mejorar el control administrativo de accesos a los archivos y otros recursos. Por ejemplo las

personas de un mismos equipo de trabajo comparten archivos comunes con lo cual tiene sentido que todos sean miembros del mismo grupo Linux. Cada grupo es identificado por una línea en el archivo /etc/group (Kerrisk, 2010).

Según la explicación de (Bovet & Cesati, 2006), los grupos son usados para compartir material selectivo con otros usuarios. Cada archivo está asociado con un grupo y se pueden otorgar privilegios distintos que a los de su propietario, por ejemplo un archivo puede estar restringido a solo lectura para el grupo.

Sumado a lo anterior, (Kerrisk, 2010) explica que existe un usuario con privilegios especiales en todo el sistema conocido como superusuario. La cuenta superusuario tiene el uid 0 y su login llamado root. Este usuario salta todas las revisiones de permisos en el sistema, además puede acceder a cualquier archivo sin importar sus privilegios. El administrador del sistema usa la cuenta superusuario para ejecutar varias tareas administrativas.

2.1.2.1.2.1. El archivo /etc/passwd

El archivo /etc/passwd, según (Kerrisk, 2010), es un archivo especial del sistema donde se almacenan todas las cuentas de usuario. Cada cuenta se guarda en una sola línea y cada línea a su vez está compuesta por 7 campos separados por el signo de dos puntos (:). En el siguiente ejemplo podemos apreciar una entrada del archivo

fids:x:1000:100:Fernando Diaz:/home/fids:/bin/bash

También explica (Kerrisk, 2010), que los campos de cada entrada son:

- ***Login name***: Es el nombre único con el cual el usuario utilizará para ingresar al sistema. A menudo es llamado username y suele ser un identificador amigable para los humanos
- ***Password***: En versiones antiguas de Linux esta campo contenia 13 caracteres encriptados, ahora esta campo es ignorado debido al uso del shadow password y en su lugar coloca la letra x para indicar que el password esta almacenado en shadow password file. Si este campo esta vacío no se pedirá el ingreso de una clave cuando se ingrese con esta cuenta.
- ***User ID (UID)***: Es el ID numérico para este usuario. Si el campo tiene el valor 0 entonces la cuenta tiene privilegios de superusuario que normalmente le pertenece al login name root. El rango aceptado en versiones antiguas (Linux 2.2 o menores) es de 16 bits osea desde 0 a 65535, en versiones Linux 2.4 y mayores, el rango es de 32 bits permitiendo hasta 2^{32}-1 uids. Es posible tambien (aunque muy inusual) tener 2 entradas en el password file con el mismo uid permitiendo que varios login names con passwords diferentes tengan acceso a los mismos recursos.
- ***Group ID (GID)***: Es el ID numérico para el primer grupo del cual el usuario es miembro. Si el usuario es miembro de otros grupos estos datos se guardarán en el archivo de grupos del sistema.
- ***Comentario***: Este campo continene un texto o comentario acerca del usuario.
- ***Home Directory***: Es el directorio inicial donde el usuario será ubicado despues del login

- ***Login Shell***: Es el programa al cual se cederá el control una vez que el usuario ingresa. Usualmente se utilizan programas denominados "shell" como bash, sh o csh pero puede ser cualquier programa. Si este campo esta vacío entonces se usara el Bourne Shell (/bin/sh) por defecto.

2.1.2.1.2.2. El archivo /etc/shadow

En los inicios de UNIX, según comenta (Jang, 2011), la seguridad no era un tema importante. Toda la información del usuario se guardaba en el archivo /etc/passwd. El problema era que cualquier usuario podía copiar ese archivo y llevarse su clave encriptada las cuales podían desencriptarse. Esto motivo el desarrollo del shadow password file (/etc/shadow) para colocar la información más sensible y ser accesible solo por el usuario root.

Los campos de cada entrada en el archivo /etc/shadow, según (Jang, 2011) son:

- ***Username***: Es el login name del usuario
- ***Password***: Contiene la contraseña encriptada del usuario
- ***Password History***: Fecha del ultimo cambio de contraseña en número de dias desde el 1 Ene 1970
- ***Min Days***: Nro mínimo de dias que el usuario debe mantener la contraseña
- ***Max Days***: Nro máximo de dias que el usuario tiene para cambiar su contraseña
- ***Warn Days***: Nro de dias de advertencia antes de la expiración de la contraseña.
- ***Inactive***: Nro de días despues de la expiración de la contraseña para que tenga la lugar la inactivación de la cuenta
- ***Disable***: Nro de días despues de la expiración de la contraseña para que tenga la lugar la deshabilitación de la cuenta

2.1.2.1.2.3. El archivo /etc/group

En las primeras implementaciones de UNIX, tal como lo cuenta (Kerrisk, 2010), un usuario solo podía pertenecer a un grupo a la vez y dicha información era guardada en el campo GID del archivo /etc/passwd. El sistema BDS 4.2 introdujo el concepto de multigrupos tiempo después y como resultado se creó el archivo /etc/group donde se almacenarían todos los grupos de los cuales el usuario era miembro. El campo GID en /etc/passwd se convertiría en lo que se conoce como grupo inicial (initial-group). Esta extraña división fue recibida y adaptada sin problemas y posteriormente estandarizada en POSIX-1-1990.

El archivo /etc/group contiene una línea por cada grupo, y cada línea está formada por 4 campos. Por ejemplo:

```
users:x:100:
admin:x:106:fids,gaby,tony,marks
```

Los campos de cada entrada, según (Kerrisk, 2010) son:

- ***Group name***: Es el nombre del grupo.
- ***Encrypted Password***: Contraseña opcional del grupo. El comando newgrp permite cambiar el initial-group de un usuario, en cuyo caso se requerirá del password de grupo si esta ha sido protegida con una contraseña. Un carácter x en este campo indica el uso del shadow password ubicado en el archivo /etc/gshadow.
- ***Group ID***: Es el ID numérico para este grupo. Normalmente existe un grupo con ID 0 llamado root
- ***User List***: Es una lista de nombres de usuarios (no uids) separada por comas

2.1.2.1.2.4. El archivo /etc/gshadow

Este archivo, afirma (Kerrisk, 2010), guarda las contraseñas de los grupos. Incluye al administrador del grupo y este archivo solo puede ser accedido por usuarios y programas con privilegios elevados.

Los campos de cada entrada, según (Jang, 2011) son:

- ***Groupname***: Es el nombre del grupo. Cada usuario tiene su propio grupo con el mismo nombre que su login. Usuarios específicos pueden ser asignados como administradores para grupos unicos.
- ***Password***: Contraseña de grupo. La mayoría de grupos tiene el carácter ! que indica que no se tiene contraseña, caso contrario se tendra una clase hash similar a la mostrada en el archivo /etc/shadow.
- ***Group ID***: Es el ID numérico del grupo asociado con el usuario
- ***Group members***: Es una lista de nombres de usuarios (no uids) separada por comas que son miembros del grupo. Si esta en blanco y el nombre del usuario es el mismo que el del grupo, se asume que ese usuario es el único miembro del grupo.

2.1.2.1.2.5. Revisión de usuarios y grupos en sistemas Linux

El sistema Linux permite a los desarrolladores tener acceso mediante la librería de funciones pwd.h y shadow.h a los registros de las cuentas de usuarios y grupos.

Para acceder a los registros del archivo /etc/passwd, (Kerrisk, 2010) enseña que, debemos usar las funciones *getpwnam*() y *getpwuid*() las cuales devuelven una estructura especial llamada "**struct passwd**". Esta estructura tiene los siguientes elementos:

```
struct passwd{
      char *pw_name;    /* Nombre de usuario */
      char *pw_passwd;  /* Contraseña */
      uid_t pw_uid;     /* User ID */
      gid_t pw_gid;     /* Group ID */
      char *pw_gecos;   /* Comentario */
      char *pw_dir;     /* Home directory */
      char *pw_shell;   /* Login shell */
};
```

Para el caso del archivo /etc/group se usan las funciones *getgrnam*() y *getgrgid*() que devuelven una estructura especial llamada "**struct group**" que tiene los siguientes elementos (Kerrisk, 2010):

```
struct group{
      char *gr_name;    /* Group name */
      char *gr_passwd;  /* Contraseña */
      gid_t gr_gid;     /* Group ID */
      char **gr_mem;    /* Lista de miembros */
};
```

Finalmente, (Kerrisk, 2010) explica que para el caso del archivo /etc/shadow, se utilizan las funciones *getspnam*() y *getspent*() las cuales devuelven la estructura "**struct spwd**" cuyos elementos son los siguientes:

```
struct spwd{
      char *sp_namp;    /* Nombre de usuario */
      char *sp_pwdp;    /* Contraseña */
      long sp_lstchg;   /* Ultimo cambio de clave */
      long sp_min;      /* Min Days */
      long sp_max;      /* Max Days */
      long sp_warn;     /* Warn Days */
      long sp_inact;    /* Inactive Days */
      long sp_expire;   /* Día de expiración */
      unsigned long sp_flag; /* Reservado */
};
```

2.1.2.1.3. Usuarios y Grupos de subsistemas Linux

"Un subsistema es un proveedor de servicio que ejecuta una o muchas funciones pero que no hace nada hasta que sea requerido" (IBM, 2013, p. 1)

Estos subsistemas según (Kerrisk, 2010), son procesos escritos para llevar a cabo tareas muy especificas y tienen las siguientes características:

- Son de larga duración, a menudo creados durante el arranque el sistema y ejecutados hasta que el sistema se detiene.
- Se ejecutan en segundo plano y sin ninguna terminal de control para asegurar su permanente actividad.

Algunos ejemplos de "subsistemas" según (Kerrisk, 2010) son los siguientes:

- ***crond***: daemon para ejecutar tareas en un tiempo planificado
- ***sshd***: daemon que permite el acceso remoto de forma segura
- ***httpd***: daemon para brindar paginas web (apache).

Algunos de estos subsistemas permiten gestionar sus propios usuarios y grupos para propósitos especiales, los cuales no estan relacionados con los usuarios y grupos del sistema operativo Linux de ninguna manera. (Jang, 2011) explica por ejemplo que el daemon httpd permite configurar contraseñas para un sitio web y que esto se consigue utilizando el programa htpasswd el cual crea una base de datos con los usuarios y contraseñas, y añade que es posible tener un grupo para brindar acceso a mas de un usuario a la vez utilizando las directivas AuthUserFile y AuthGroupFile según sea el caso.

Un daemon de importancia para este proyecto es mysqld, que corresponde al motor de base de datos de la empresa Oracle, el cual permite tambien gestionar sus propios usuarios.

2.1.2.1.3.1. Usuarios del subsistema MySQL

MySQL según (Shasankar, 2013), es la aplicación de base de datos open source más usada en el mundo. Está disponible para individuos y empresas que desean desarrollar sus páginas web y aplicaciones usando la base de datos MySQL.

Las cuentas de MySQL no son como las cuentas Linux, ya que MySQL considera el origen de un intento de login como parte del proceso de autenticación. Es decir, un usuario MySQL está compuesto por el login y su ubicación (hostname, ip o un wildcard). Así pues, el usuario fids que se conecta desde el ip 127.0.0.1 es diferente al usuario fids que se conecta desde el host demo.com, manteniendo credenciales y privilegios por separado (Schwartz, Zaitsev, Tkachenko, Zawodny, Lentz, & Balling, 2008).

Por su parte, (MySQL-AB, 2006) refiere que la función principal del sistema de privilegios de MySQL es autenticar a los usuarios que se conectan desde un host determinado, y asociar a ese usuario con privilegios de una o más bases de datos. El control de acceso de MYSQL se produce en 2 fases cuando un programa cliente conecta al servidor:

- *Fase 1*: El servidor valida la identidad del usuario y acepta o rechaza la conexión según los datos enviados.
- *Fase 2*: Una vez conectado, el sistema valida cada sentencia ejecutada para determinar si se tienen los privilegios suficientes para permitirla.

Los usuarios y sus privilegios se almacenan en la base de datos "**mysql**" concretamente en las tablas "*user*", "*host*", "*db*", "*tables_priv*", "*column_priv*" y "*procs_priv*". De todas ellas, la tabla "user" es la única que se utiliza para la Fase 1 del proceso de autenticación. Luego, durante la Fase 2, intervienen todas las tablas (incluso la tabla "user") para validar los privilegios; La tabla "user" es una tabla especial ya que permite definir privilegios que se aplican a todas las bases de datos. (MySQL-AB, 2006).

Los privilegios que se pueden otorgar o revocar en el sistema MySQL son identificados mediante los datos guardados en columnas especiales, que pueden encontrarse en una o varias de las tablas de privilegios. (MySQL-AB, 2006). A continuación se detallan los privilegios y sus funciones asociadas.

Tabla 7

Privilegios MySQL

Privilegio	**Columna**	**Contexto**	**Permite**
CREATE	Create_priv	db, tbl, idx	Crear nuevas bases de datos y tablas
CREATE TEMPORARY TABLES	Create_tmp_table_priv	tbl	Crear tablas temporales
CREATE VIEW	Create_view_priv	view	Crear vistas
CREATE ROUTINE	Create_routine_priv	routines	Crear procedimientos y funciones
CREATE USER	Create_user_priv	admin	Crear, eliminar o modificar usuarios
DROP	Drop_priv	db, tbl, view	Eliminar base de datos, tablas o

			vistas
SELECT	Select_priv	tbl, col	Leer registros de una tabla
INSERT	Insert_priv	tbl, col	Insertar registros en las tablas
DELETE	Delete_priv	tbl	Eliminar registros
UPDATE	Update_priv	tbl, col	Actualizar registros
ALTER	Alter_priv	tbl	Cambiar la estructura de la tabla. Requiere CREATE, INSERT y DROP
ALTER ROUTINE	Alter_routine_priv	routines	Cambiar o eliminar rutinas
EXECUTE	Execute_priv	routines	Ejecutar rutinas
SHOW VIEW	Show_view_priv	view	Mostrar la definición de vistas
INDEX	Index_priv	tbl	Crear índices en tablas creadas
TRIGGER	Trigger_priv	tbl	Operar con desencadenadores
EVENT	Event_priv	db	Crear, eliminar y manipular eventos
REFERENCES	References_priv	db, tbl	Aplicar restricciones de clave foránea
LOCK TABLES	Lock_tables_priv	db	Bloquear una tabla con privilegio SELECT
GRANT OPTION	Grant_priv	db, tbl, routines	Otorgar o revocar a otros usuarios los privilegios que uno tiene
USAGE		admin	Dejar sin ningún privilegio a un usuario
ALL		admin	Otorgar todos los

[PRIVILEGES]			privilegios según contexto a excepción de GRANT OPTION
SUPER	Super_priv	admin	Usar funciones avanzadas de administración
SHOW DATABASES	Show_db_priv	admin	Ver los nombres de todas las bases de datos
SHUTDOWN	Shutdown_priv	admin	Usar el comando mysqladmin shutdown
RELOAD	Reload_priv	admin	Realizar operaciones FLUSH
PROCESS	Process_priv	admin	Usar los comandos SHOW PROCCESLIST y SHOW ENGINE
FILE	File_priv	server	Leer y escribir archivos en el host
REPLICATION CLIENT	Repl_client_priv	admin	Usar los comandos SHOW MASTER STATUS, SHOW SLAVE STATUS y SHOW BINARY LOGS
REPLICATION SLAVE	Repl_slave_priv	admin	Conectar al servidor maestro y recibir actualizaciones

Fuente: (MySQL-AB, 2006) sección 4.8

Donde el campo contexto puede ser:

- *db*: Se aplica a una base de datos del sistema o a todas.
- *tbl*: Se aplica a las tablas de una base de datos o a todas.

- *idx*: Se aplica al índice de una tabla.
- *view*: Se aplica a una vista en una base de datos
- *routines*: Se aplica a un procedimiento almacenado o función
- *admin:* Se aplican a tareas administrativas en la operación del servidor de base de datos.
- *server:* Se aplica al servidor Linux donde reside el servidor de base de datos MySQL.

Para tener un panorama más claro, se revisará la estructura de las tablas de privilegios. Esta estructura puede ser revisada siguiendo las siguientes instrucciones en un servidor Linux.

```
[root@server ~]# service mysqld start
[root@server ~]# mysql -u root -p
mysql> describe mysql.user;
```

La estructura mostrada para la tabla mysql.user se explica a continuación:

Tabla 8

Estructura de la tabla mysql.user

Campo	**Tipo de Datos**	**Descripción**
Host	char(60) PK	Host del usuario
User	char(16) PK	Nombre del usuario
Password	char(41)	Contraseña del usuario
Select_priv	enum('N','Y')	Permiso SELECT global
Insert_priv	enum('N','Y')	Permiso INSERT global
Update_priv	enum('N','Y')	Permiso UPDATE global
Delete_priv	enum('N','Y')	Permiso DELETE global
Create_priv	enum('N','Y')	Permiso CREATE global
Drop_priv	enum('N','Y')	Permiso DROP global
Reload_priv	enum('N','Y')	Permiso RELOAD
Shutdown_priv	enum('N','Y')	Permiso SHUTDOWN
Process_priv	enum('N','Y')	Permiso PROCESS
File_priv	enum('N','Y')	Permiso FILE
Grant_priv	enum('N','Y')	Permiso GRANT global
References_priv	enum('N','Y')	Permiso REFERENCES global

Index_priv	enum('N','Y')	Permiso INDEX global
Alter_priv	enum('N','Y')	Permiso ALTER global
Show_db_priv	enum('N','Y')	Permiso SHOW DATABASES
Super_priv	enum('N','Y')	Permiso SUPER
Create_tmp_table_priv	enum('N','Y')	Permiso CREATE TEMP TABLE global
Lock_tables_priv	enum('N','Y')	Permiso LOCK TABLES global
Execute_priv	enum('N','Y')	Permiso EXECUTE global
Repl_slave_priv	enum('N','Y')	Permiso REPLICATION SLAVE
Repl_client_priv	enum('N','Y')	Permiso REPLICATION CLIENT
Create_view_priv	enum('N','Y')	Permiso CREATE VIEW global
Show_view_priv	enum('N','Y')	Permiso SHOW VIEW global
Create_routine_priv	enum('N','Y')	Permiso CREATE ROUTINE global
Alter_routine_priv	enum('N','Y')	Permiso ALTER ROUTINE global
Create_user_priv	enum('N','Y')	Permiso CREATE USER
Event_priv	enum('N','Y')	Permiso EVENT
Trigger_priv	enum('N','Y')	Permiso TRIGGER
ssl_type	enum('','ANY', 'X509', 'SPECIFIED')	Tipo TLS que debe usar la cuenta
ssl_cipher	blob	Cifrado TLS de la cuenta
x509_issuer	blob	Cifrado x509 de la cuenta
x509_subject	blob	Asunto x509 de la cuenta
max_questions	int(11)	Cantidad máxima de consultas por hora de la cuenta.
max_updates	int(11)	Cantidad máxima de actualizaciones por hora de la cuenta.
max_connections	int(11)	Cantidad máxima de conexiones que una cuenta puede iniciar en una hora.
max_user_connections	int(11)	Cantidad máxima de conexiones que una cuenta puede tener en simultáneo.

Fuente: (MySQL-AB, 2006) sección 4.8

La estructura mostrada para la tabla mysql.db se explica a continuación:

Tabla 9

Estructura de la tabla mysql.db

Campo	Tipo de Datos	Descripción
Host	char(60) PK	Host del usuario
Db	char(64) PK	Nombre de la base de datos
User	char(16) PK	Nombre del usuario
Select_priv	enum('N','Y')	Permiso SELECT
Insert_priv	enum('N','Y')	Permiso INSERT
Update_priv	enum('N','Y')	Permiso UPDATE
Delete_priv	enum('N','Y')	Permiso DELETE
Create_priv	enum('N','Y')	Permiso CREATE
Drop_priv	enum('N','Y')	Permiso DROP
Grant_priv	enum('N','Y')	Permiso GRANT
References_priv	enum('N','Y')	Permiso REFERENCES
Index_priv	enum('N','Y')	Permiso INDEX
Alter_priv	enum('N','Y')	Permiso ALTER
Create_tmp_table_priv	enum('N','Y')	Permiso CREATE TEMP TABLE
Lock_tables_priv	enum('N','Y')	Permiso LOCK TABLES
Create_view_priv	enum('N','Y')	Permiso CREATE VIEW
Show_view_priv	enum('N','Y')	Permiso SHOW VIEW
Create_routine_priv	enum('N','Y')	Permiso CREATE ROUTINE
Alter_routine_priv	enum('N','Y')	Permiso ALTER ROUTINE
Execute_priv	enum('N','Y')	Permiso EXECUTE
Event_priv	enum('N','Y')	Permiso EVENT
Trigger_priv	enum('N','Y')	Permiso TRIGGER

Fuente: (MySQL-AB, 2006) sección 4.8

La estructura mostrada para la tabla mysql.host se explica a continuación:

Tabla 10

Estructura de la tabla mysql.host

Campo	Tipo de Datos	Descripción
Host	char(60) PK	Host del usuario
Db	char(64) PK	Nombre de la base de datos
Select_priv	enum('N','Y')	Permiso SELECT

Insert_priv	enum('N','Y')	Permiso INSERT
Update_priv	enum('N','Y')	Permiso UPDATE
Delete_priv	enum('N','Y')	Permiso DELETE
Create_priv	enum('N','Y')	Permiso CREATE
Drop_priv	enum('N','Y')	Permiso DROP
Grant_priv	enum('N','Y')	Permiso GRANT
References_priv	enum('N','Y')	Permiso REFERENCES
Index_priv	enum('N','Y')	Permiso INDEX
Alter_priv	enum('N','Y')	Permiso ALTER
Create_tmp_table_priv	enum('N','Y')	Permiso CREATE TEMP TABLE
Lock_tables_priv	enum('N','Y')	Permiso LOCK TABLES
Create_view_priv	enum('N','Y')	Permiso CREATE VIEW
Show_view_priv	enum('N','Y')	Permiso SHOW VIEW
Create_routine_priv	enum('N','Y')	Permiso SHOW VIEW
Alter_routine_priv	enum('N','Y')	Permiso CREATE ROUTINE
Execute_priv	enum('N','Y')	Permiso ALTER ROUTINE
Trigger_priv	enum('N','Y')	Permiso TRIGGER

Fuente: (MySQL-AB, 2006) sección 4.8

La estructura mostrada para la tabla mysql.tables_priv se explica a continuación:

Tabla 11

Estructura de la tabla mysql.tables_priv

Campo	**Tipo de Datos**	**Descripción**
Host	char(60) PK	Host del usuario
Db	char(64) PK	Nombre de la base de datos
User	char(16) PK	Nombre del usuario
Table_name	char(64) PK	Nombre de la tabla
Grantor	char(77) FK	Usuario que asignó el permiso
Timestamp	timestamp	Fecha de registro
Table_priv	set('Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant',	Posibles permisos de la tabla

	'References', 'Index', 'Alter', 'Create View', 'Show view', 'Trigger')	
Column_priv	set('Select', 'Insert', 'Update', 'References')	Posibles permisos de la columna

Fuente: (MySQL-AB, 2006) sección 4.8

La estructura mostrada para la tabla mysql.columns_priv se explica a continuación:

Tabla 12

Estructura de la tabla mysql.columns_priv

Campo	**Tipo de Datos**	**Descripción**
Host	char(60) PK	Host del usuario
Db	char(64) PK	Nombre de la base de datos
User	char(16) PK	Nombre del usuario
Table_name	char(64) PK	Nombre de la tabla
Column_name	char(64) PK	Nombre de la columna
Timestamp	timestamp	Fecha de registro
Column_priv	set('Select', 'Insert', 'Update', 'References')	Posibles permisos de la columna

Fuente: (MySQL-AB, 2006) sección 4.8

La estructura mostrada para la tabla mysql.procs_priv se explica a continuación:

Tabla 13

Estructura de la tabla mysql.proc_priv

Campo	**Tipo de Datos**	**Descripción**
Host	char(60) PK	Host del usuario
Db	char(64) PK	Nombre de la base de datos
User	char(16) PK	Nombre del usuario
Routine_name	char(64) PK	Nombre de la rutina

Routine_type	enum('FUNCTION', 'PROCEDURE') PK	Tipo de rutina
Grantor	char(77) FK	Usuario que asignó el permiso
Proc_priv	set('Execute', 'Alter Routine', 'Grant')	Posibles permisos de la rutina
Timestamp	timestamp	Fecha de registro

Fuente: (MySQL-AB, 2006) sección 4.8

2.1.2.2. ISO/IEC 27002:2013

El contenido descrito en esta sección y en todas sus subsecciones ha sido obtenido mediante una traducción personal del autor del documento estándar internacional “ISO/IEC 27002 - Information technology - Security techniques - Code of practice for information security controls” (ISO/IEC, 2013) como parte del proceso de investigación de la presente tesis.

Técnicas de Seguridad de Tecnologías de la Información – Mejores prácticas para los controles de la seguridad de la información

Según podemos encontrar en la (ISO/IEC, 2013); ISO (the International Organization for Standardization) y la IEC (the International Electrotechnical Commission) forman el sistema especializado de estandarización a nivel mundial. Organismos nacionales que son miembros de la ISO o IEC participan en el desarrollo de estándares internacionales a través de comités técnicos establecidos por la respectiva organización para lidiar con campos particulares de actividad técnica.

“Los comités técnicos ISO e IEC colaboran en campos de interés junto a organizaciones internacionales, y gubernamentales. En el campo de las TIC, tienen establecido un comité técnico conjunto, la ISO/IEC JTC 1, quien preparo la ISO/IEC 27002”. (ISO/IEC, 2013).

2.1.2.2.1. Introducción

a) Trasfondo y Contexto

Según la (ISO/IEC, 2013), este estándar internacional está diseñado por organizaciones para ser usado como referencia en la selección de controles al momento de implementar un SGSI (Sistema de Gestión de Seguridad de la Información) basado en la ISO/IEC 27001, o como un

documento guía para organizaciones que se encuentren implementando controles de seguridad mundialmente aceptados.

Así mismo, la (ISO/IEC, 2013) explica que las organizaciones de todos los tipos y tamaños (incluyendo el sector público y privado, comerciales o sin fines de lucro) recopilan, procesan, guardan y transmiten información de diversas maneras, las cuales incluyen: medios electrónicos, físicos y verbales (p.e: conversaciones y presentaciones).

El valor de la información según la (ISO/IEC, 2013) va más allá de las palabras escritas, números e imágenes; algunos ejemplos de información intangible son: conocimiento, conceptos, ideas. En un mundo interconectado, la información y sus procesos relacionados, los sistemas, las redes y el personal que participan en su funcionamiento, el manejo y la protección son activos que, al igual que otros activos comerciales de importancia, son valiosas para los negocios y la organización y en consecuencia merecen o requieren de protección contra diversos riesgos.

En su explicación, la (ISO/IEC, 2013) detalla que los activos están sujetos a amenazas tanto deliberadas como accidentales mientras que los procesos, sistemas, redes y personas tienen vulnerabilidades inherentes. Los cambios en los procesos de negocio y sistemas u otros cambios externos (por ejemplo, nuevas leyes y reglamentos) pueden crear nuevos riesgos de seguridad de la información. Es por ello, que teniendo en cuenta las diversas formas en que las amenazas pueden aprovecharse de las vulnerabilidades para dañar la organización, los riesgos de seguridad de la información están siempre presentes. La Seguridad de la información eficaz reduce estos riesgos mediante la protección de la organización contra las amenazas y vulnerabilidades, y luego reduce impactos a sus activos.

La seguridad de la información se logra según la (ISO/IEC, 2013), mediante la implementación de un conjunto adecuado de controles, incluyendo políticas, procesos, procedimientos, estructuras organizativas y funciones de software y hardware. Dichos controles deben ser establecidos, implementados, supervisados, revisados y mejorados

cuando sea necesario, para asegurar que se cumplen los objetivos específicos de seguridad y de negocios en la organización. Un SGSI como el señalado en la ISO/IEC 27001 tiene una perspectiva coordinada e integral de los riesgos de seguridad de información en la organización para implementar un conjunto de controles muy completo que aseguren la información bajo el marco general de un sistema de gestión coherente.

La realidad hoy en día para la (ISO/IEC, 2013), es que muchos sistemas de información no han sido diseñados para ser seguros, ya que no cumplen los lineamientos de la ISO/IEC 27001 ni el estándar (ISO/IEC 27002). La seguridad que se puede lograr a través de medios técnicos es limitada y debe ser apoyado por la administración y procedimientos apropiados. Se requiere de una planificación cuidadosa y una atención a los detalles para identificar que controles debe incluirse. Un SGSI exitoso requiere el constante apoyo de todos los empleados en la organización. También puede requerirse la participación de los proveedores, accionistas u otras partes externas. El asesoramiento especializado también puede ser necesario.

b) Requerimientos de seguridad de la información

Es indispensable según la (ISO/IEC, 2013), que una organización identifique sus requerimientos de seguridad. Para ello existen tres ramas de requerimientos y son los siguientes:

- ✓ La evaluación de los riesgos de la organización, teniendo en cuenta las estrategias y los objetivos generales de la organización. A través de una evaluación de riesgos, se identifican las amenazas a los activos, la vulnerabilidad y su probabilidad de ocurrencia y se estima el impacto potencial
- ✓ Los requisitos legales, de estatutos, requerimientos regulatorios y contractuales que una organización junto a sus socios comerciales, contratistas y proveedores de servicios tienen que satisfacer
- ✓ El conjunto de objetivos, principios y requisitos de negocio para el manejo de la información, procesamiento, almacenaje, comunicación y archivado que una organización ha construido para apoyar sus operaciones.

Durante la implementación de los controles, la (ISO/IEC, 2013) sugiere que los recursos utilizados deben equilibrarse con el daño probable al negocio ocasionado por la ausencia de esos controles de seguridad. Los resultados que se obtienen en una evaluación de riesgos nos ayudan a guiar y así mismo determinar la acción que sea más adecuada a la gestión y prioridades para manejar sus riesgos de seguridad así como la implementación de controles elegidos para proteger dichos riesgos.

Según encontramos en la (ISO/IEC, 2013), el documento ISO/IEC 27005 provee una guía de gestión de riesgos de seguridad de la información, la cual incluye la asesoría sobre su evaluación, tratamiento, aceptación, comunicación, seguimiento y la revisión de riesgos.

c) Eligiendo controles

Los controles según la (ISO/IEC, 2013), pueden elegirse de esta norma o de otros conjuntos de controles. También pueden ser diseñados nuevos controles para satisfacer las necesidades específicas, según corresponda.

Es la organización quien toma la decisión de elegir sus controles, conforme a lo explicado por la (ISO/IEC, 2013), esta elección está basada en los criterios de aceptación de riesgos, junto a las opciones de tratamiento del riesgo, al enfoque de gestión del riesgo general que se aplica a la organización, y también debe estar sujeta a todas las leyes, reglamentos nacionales e internacionales pertinentes. La elección de estos controles depende de la manera en que interactúan para proporcionar la defensa en profundidad.

La (ISO/IEC, 2013) menciona que algunos de los controles en esta norma pueden ser considerados como principios rectores para la gestión de la seguridad de la información y es aplicable para la mayoría de las organizaciones.

d) Desarrollo de sus propias guías

Esta norma puede considerarse según la (ISO/IEC, 2013), como un punto de partida para el desarrollo de las guías específicas para la organización. No todos los controles en este código de prácticas pueden ser aplicables. Por otra parte, pueden ser necesarios controles y directrices adicionales no incluidos en esta norma. Cuando los documentos son desarrollados conteniendo directrices o controles adicionales, puede ser útil incluir referencias cruzadas a las cláusulas en esta norma cuando sea aplicable, para facilitar la labor de los auditores y socios comerciales.

e) Consideraciones del ciclo de vida

La (ISO/IEC, 2013), nos explica que la información tiene un ciclo de vida natural, desde la creación y emisión hasta el almacenamiento, uso, procesamiento y transmisión hasta su eventual destrucción o deterioro. El valor de los activos y sus riesgos pueden variar durante el curso de su vida, pero la seguridad de la información continua siendo sustancial hasta cierto punto en todas las etapas.

En este contexto, la (ISO/IEC, 2013) manifiesta que los sistemas de información tienen ciclos de vida dentro de las cuales se conciben, se especifican, se diseñan, se desarrollan, se prueban, se implementan, se utilizan, se mantienen y finalmente se retiran del servicio. La seguridad de la información debe ser tomada en cuenta en cada etapa. Las organizaciones pueden mejorar sus controles de seguridad al desarrollar nuevos sistemas o cambiar los ya existentes.

f) Normas relacionadas

Según lo explica la (ISO/IEC, 2013), a pesar de la gran cantidad de controles de seguridad ofrecidos por esta norma, y que son aplicados en muchas organizaciones, la familia de estándares ISO/IEC 27000 proporciona asesoría complementaria en otros aspectos del proceso general de la gestión de seguridad de la información.

2.1.2.2.2. Ambito

El ámbito de este estándar internacional indicado por la (ISO/IEC, 2013), incluye la selección, implementación y gestión de los controles, teniendo en

cuenta el entorno de riesgo de la seguridad de la información de la organización.

Este estándar internacional está diseñado según la (ISO/IEC, 2013), para ser usado por organizaciones que deseen:

- ✓ Seleccionar adecuadamente los controles en su proceso de implementación de un SGSI basado en la ISO/IEC 27001
- ✓ Implementar controles de seguridad mundialmente aceptados
- ✓ Elaborar guías de gestión propias referentes a la seguridad de la información

2.1.2.2.3. Estructura

La estructura de este documento tal como lo explica la (ISO/IEC, 2013), contiene 14 dominios o cláusulas de controles de seguridad y un total de 35 categorías de seguridad principales y 114 controles.

a) Cláusulas

Las cláusulas según la (ISO/IEC, 2013), definen los controles de seguridad que contiene una o más categorías principales de seguridad, y su orden no tiene importancia. Dependiendo de las circunstancias, estos controles de cualquiera o todas las cláusulas podrían ser importantes; por lo tanto, cada organización al aplicar esta norma debería identificar los controles aplicables, lo importante que son y su aplicación a los procesos de negocio individuales.

b) Categorías de Control

Cada categoría principal de control de seguridad según la (ISO/IEC, 2013), contiene lo siguiente:

- ✓ Un objetivo de control que indica lo que se quiere lograr
- ✓ Uno o más controles que se pueden aplicar para alcanzar el objetivo de control

Tal y como nos indica la (ISO/IEC, 2013), las descripciones de control están estructuradas de la siguiente manera: control, guía de implementación e información adicional.

Control

Según la (ISO/IEC, 2013), en el control se define la sentencia de control específica, para lograr los estándares del objetivo de control.

Guía de implementación

La (ISO/IEC, 2013) explica que, la guía de implementación brinda información en detalle que sirve de apoyo a la implementación del control y su cumplimiento. La guía puede ser insuficiente en algunos escenarios o no cumplir con los requerimientos de control específicos de una organización.

Información adicional

Por último, la información adicional según la (ISO/IEC, 2013), proporciona información complementaria que pueda ser de utilidad, entre ellas: aspectos legales y referencias a otras normativas. De no haber información adicional, esta sección puede omitirse.

2.1.2.2.4. Dominio Control de Accesos

a) Requisitos de negocio para el control de accesos (Control 9.1)

Objetivo: Limitar el acceso a la información y a las instalaciones en las que se procesan.

1. Política de control de acceso (Control 9.1.1)

Control

Se debe establecer una política de control de acceso, la cual debe ser revisada y documentada en base al negocio y sus requerimientos de seguridad de la información.

Guía de Implementación

Las normas para el control de acceso, derechos y restricciones en las funciones de usuarios en relación con los activos de la institución, a criterio de la (ISO/IEC, 2013), deben ser determinadas por sus

propietarios, las cuales deben ir acorde con tal nivel y rigor de detalle que reflejen los riesgos de seguridad de la información asociada.

Así mismo, la (ISO/IEC, 2013) nos explica que los controles de acceso deben ser considerados como lógicos y físicos en conjunto. Los usuarios y proveedores deben tener claro los requerimientos del negocio sobre estos controles de acceso para poder cumplirlos.

Según la (ISO/IEC, 2013), la política debe tener en cuenta lo siguiente

a. Requisitos de seguridad de las aplicaciones de negocio.
b. Políticas para la difusión de la información y su autorización, por ejemplo, los niveles de seguridad y clasificación de la información.
c. La coherencia que existe entre los derechos de acceso y políticas de clasificación de la información de los sistemas y redes.
d. Legislación relevante y las obligaciones contractuales respecto de límite de acceso a los datos o servicios.
e. Administración de los derechos de acceso en un entorno distribuido e interconectado que reconoce todo tipo de conexiones disponibles.
f. Separación de roles de control de acceso, tales como la solicitud, autorización, y la administración de accesos.
g. Requerimientos para la autorización formal de las solicitudes de acceso.
h. Requerimientos para la revisión recurrente de los derechos de acceso
i. Eliminación de los derechos de acceso.
j. Registro de todos los eventos relevantes sobre el uso y administración de credenciales de usuarios.
k. Perfiles con acceso privilegiado.

Información adicional

Como información adicional, la (ISO/IEC, 2013) aclara que, cuando se especifican las reglas de control de acceso, se debe considerar:

a. Establecer reglas basadas en la premisa "Todo es denegado a menos que se permita expresamente" en lugar de una regla más débil como "Todo es permitido a menos que se prohíba expresamente".

b. Los cambios en las etiquetas de información que se inicien automáticamente por instalaciones de procesamiento de información y así mismo las iniciadas a petición de un usuario.
c. Los cambios realizados en los permisos de usuario que son iniciados de manera automática por el sistema de información y aquellos que son ejecutados por un administrador.
d. Las reglas que requieren aprobación de manera específica antes de su ejecución.

Es importante considerar, según la (ISO/IEC, 2013), que las reglas de control de acceso necesitan ser respaldadas tanto por procedimientos formales como por responsabilidades definidas. El control de acceso basado en roles se define como una estrategia que ha dado buenos resultados en muchas organizaciones para relacionar los derechos de acceso con los perfiles del negocio.

Entre los principios más frecuentes, encontrados por la (ISO/IEC, 2013), que guían la política de control de acceso, tenemos:

a. Información necesaria: se brinda acceso sólo a la información mínima necesaria para poder realizar sus actividades (diferentes actividades o perfiles requieren información necesaria distinta y por lo tanto distintos perfiles de acceso).
b. Medios necesarios: se brinda acceso sólo a los medios de procesamiento de información necesarios para poder realizar sus actividades (equipos de cómputo, aplicaciones, centro de datos, etc).

2. Acceso a redes y servicios de red (Control 9.1.2)

Control

Los usuarios deben acceder únicamente a la red y sus servicios a los que han sido específicamente autorizados a utilizar.

Guía de Implementación

Para acceder a las redes y sus servicios, la (ISO/IEC, 2013) recomienda que, se debe formular una política que se encuentre en relación con el uso de las redes y sus servicios, dicha política debe cubrir lo siguiente:

a. Las redes y sus servicios a los que estén autorizados acceder.
b. Procedimientos de autorización que permitan saber quién puede acceder a que redes y a cuales de sus servicios.
c. Procedimientos y controles de gestión para proteger el acceso a las conexiones de red y sus servicios.
d. Los medios usados para acceder a las redes y sus servicios (e.g. uso de redes VPN o redes inalámbricas)
e. Requerimientos de autenticación de usuario para el acceso a varios servicios de red.
f. El seguimiento (monitoreo) de la utilización de los servicios de red.

Las políticas de la organización concernientes al uso de los servicios de red y el control de acceso deben ser coherentes.

Información adicional

De acuerdo a la (ISO/IEC, 2013), todas las conexiones a los servicios de red que no estén autorizadas o sean inseguras, pueden comprometer a toda la organización. Este control es especialmente importante para las conexiones de red a las aplicaciones críticas de negocios o para los usuarios que se encuentren en lugares de alto riesgo, como por ejemplo, las áreas públicas o externas que estén fuera del control y administración del área de seguridad de la información en la organización.

b) Gestión de acceso a usuarios (Control 9.2)

Objetivo: Asegurar el acceso de los sistemas y servicios a los usuarios autorizados y prevenir los accesos a usuarios no autorizados.

1. **Registro y eliminación de usuarios (Control 9.2.1)**

   Control

   Se debe implementar un proceso formal de registro y eliminación de usuarios para permitir la asignación de derechos de acceso.

   Guía de Implementación

   En esta guía, la (ISO/IEC, 2013) sostiene que el proceso para gestionar identificadores (ID) de usuarios debe incluir:

   a. El uso de identificadores de usuarios únicos (IDs) que permitan a los usuarios estar vinculados y ser responsables de sus actividades; el uso de identificadores compartidos sólo se permitirá cuando sean necesarias por razones operacionales o de negocio y debe ser aprobado y documentado.
   b. Deshabilitar o eliminar inmediatamente los ID de los usuarios que han abandonado la organización.
   c. Identificar o eliminar periódicamente los ID de usuarios redundantes.
   d. Asegurarse de que los ID de usuario redundantes no sean asignados a otros usuarios.

   Información adicional

   Según la (ISO/IEC, 2013), existen 2 pasos para otorgar o revocar el acceso ya sea a la información o a las instalaciones donde se procesa, y son:

   a. La asignación y habilitación o la revocación de un ID de usuario.
   b. Otorgar o revocar, los derechos de acceso a los ID de usuario mencionados.

2. **Distribución de acceso a usuarios (Control 9.2.2)**

   Control

   Se debe implementar un proceso formal de distribución de acceso a usuarios para otorgar o revocar derechos de acceso a todos los tipos de usuarios en todos los sistemas y servicios.

Guías de Implementación

El proceso de distribución para asignar o revocar los derechos de acceso otorgados a los ID de los usuarios según la (ISO/IEC, 2013), debe incluir:

a. Contar con la autorización del propietario del sistema de información o servicio para su uso; se puede requerir además, la aprobación para los derechos de acceso de gestión.
b. Comprobar que el nivel de acceso concedido es adecuado a las políticas de acceso y es consistente con otros requisitos como la separación de funciones.
c. Garantizar que los derechos de acceso no estén activados (e.g. por proveedores de servicios) antes que los procedimientos de autorización estén terminados.
d. Mantener una bitácora central de los derechos de acceso concedido a un ID de usuario para acceder a los sistemas de información y servicios.
e. Actualizar los derechos de acceso de los usuarios que han cambiado los roles o puestos de trabajo e inmediatamente retirar o bloquear los derechos de acceso de los usuarios que han abandonado la organización.
f. Revisar de manera periódica los derechos de acceso junto a los propietarios de los sistemas de información o servicios.

Información adicional

Se debe considerar, a sugerencia de la (ISO/IEC, 2013), la posibilidad de establecer los roles de acceso de usuario basándose en los requerimientos del negocio que resumen una serie de derechos de acceso en perfiles típicos de acceso de usuario. Las solicitudes de acceso y las revisiones se gestionan más fácilmente a nivel de roles que a nivel de derechos particulares.

Es necesario considerar, a criterio de la (ISO/IEC, 2013), la posibilidad de incluir cláusulas en los contratos del personal y contratos de servicios que especifiquen las sanciones en caso el personal o contratistas intenten conseguir accesos no autorizados.

3. **Gestión de derechos de acceso privilegiados (Control 9.2.3)**

Control

Se debe restringir y controlar la asignación y uso de los derechos de acceso privilegiados.

Guía de Implementación

Según la (ISO/IEC, 2013), la asignación de derechos de acceso privilegiados debe ser controlada a través de un proceso de autorización formal en conformidad con la política de control de acceso correspondiente. Los siguientes pasos se deben tener en cuenta:

a. Los derechos de acceso privilegiados que son asociados a cada sistema o proceso, (e.g. sistema operativo, sistema gestor de base de datos) así como a cada aplicación y a los usuarios a los que tienen que otorgarles dichos accesos deben ser identificados.
b. Los derechos de acceso privilegiados deben asignarse en base a los principios de “medios necesarios” y de “evento por evento” de acuerdo a la política de control de acceso, por ejemplo, basándose en el requisito mínimo por sus roles de función.
c. Se debe mantener un proceso de autorización y registro de todos los privilegios asignados. Los derechos de acceso privilegiado no deben otorgarse hasta que el proceso de autorización se haya completado.
d. Se debe definir los requisitos para la caducidad de los derechos de acceso privilegiados.
e. Los derechos de acceso privilegiados se deben otorgar a un ID de usuario distinto de los utilizados para actividades comunes del negocio. Las actividades comunes del negocio no deben realizarse con ID de usuarios privilegiados.
f. Las facultades de los usuarios con derechos de acceso privilegiados se deben revisar de manera periódica a fin de verificar si están de acuerdo con sus funciones.
g. Para evitar el uso no autorizado de los ID genéricos de usuarios administrativos, se deben establecer y mantenerse procedimientos específicos de acuerdo a las capacidades de configuración de los sistemas.

h. La privacidad de la información de autenticación para los ID genéricos de usuarios administrativos debe ser mantenida cuando se comparten (por ejemplo, cambiar las contraseñas con frecuencia y tan pronto como sea posible cuando un usuario privilegiado deja o cambia de trabajo).

Información adicional

Para la (ISO/IEC, 2013), el uso inadecuado de los privilegios de administración del sistema, es decir, cualquier característica de un sistema de información que permita al usuario sobrescribir controles a los sistemas o aplicaciones, es una de las causas más comunes que genera fallas o brechas de seguridad en los sistemas.

4. **Gestión de información confidencial de autenticación de usuarios (Control 9.2.4)**

Control

Se debe controlar la asignación de información confidencial de autenticación a través de un proceso de gestión formal.

Guía de Implementación

Para esta guía, la (ISO/IEC, 2013) indica que el proceso debe incluir los siguientes requerimientos:

a. Se debe solicitar a los usuarios la firma de una declaración para que la información de autenticación personal sea confidencial, así como la información grupal (e.g. información compartida) la cual, solo debe ser compartida con los miembros del grupo; esta declaración se puede incluir en los términos y condiciones de empleo.
b. Cuando los usuarios son los encargados de mantener su propia información de autenticación, deben ser provistos inicialmente con información de autenticación temporal, la cual deben cambiar de forma obligatoria en su primer uso.
c. Se deben establecer procedimientos para verificar la identidad de un usuario antes de proporcionarle información de autenticación nueva, temporal o de reemplazo.

d. La información de autenticación temporal se debe dar a los usuarios de una manera segura; el uso de agentes externos o mensajes de correo electrónico no protegidos (texto plano) deben ser evitados.
e. La información de autenticación temporal debe ser única para un individuo y no debe ser predecible.
f. Los usuarios deben confirmar la recepción de la información de autenticación.
g. La información de autenticación por defecto de los sistemas o software de terceros debe ser modificada durante su instalación.

Información adicional

Las contraseñas, según la (ISO/IEC, 2013), son un tipo de información de autenticación de uso común y un medio habitual para verificar la identidad de un usuario. Otros tipos de información de autenticación son las claves criptográficas y otros datos almacenados en tokens vía hardware (e.g. tarjetas inteligentes) que producen códigos de autenticación.

5. Revisión de los derechos de acceso a los usuarios (Control 9.2.5)

Control

Los encargados de los activos deben revisar los derechos de acceso de los usuarios periódicamente.

Guías de Implementación

Esta revisión de derechos de acceso, según la (ISO/IEC, 2013), debe considerar lo siguiente:

a. Los derechos de acceso de los usuarios deben ser revisados periódicamente y después de cualquier cambio, como la promoción, degradación o el cese definitivo.
b. Los derechos de acceso de usuario deben ser revisados y reasignados al pasar de un rol a otro dentro de la misma organización.
c. Las autorizaciones a los derechos de acceso privilegiados deben revisarse en intervalos más frecuentes.
d. Las asignaciones de privilegios deben comprobarse periódicamente para garantizar que no se han conseguido privilegios no autorizados.

e. Cualquier cambio en cuentas privilegiadas debe ser registrado para su revisión periódica.

Información adicional

Este control, según la (ISO/IEC, 2013), subsana las posibles debilidades en la ejecución de los controles 9.2.1, 9.2.2 y 9.2.6.

6. Eliminación o ajuste de los derechos de acceso (Control 9.2.6)

Control

Se deben retirar los derechos de acceso a la información e instalaciones de procesamiento de información, a todos los empleados y usuarios externos al término de su empleo, acuerdo, contrato o deben ser modificadas en caso de ocurrir algún cambio.

Guía de Implementación

En caso de cese, la (ISO/IEC, 2013) sostiene, que los derechos de acceso de una persona a la información, servicios y activos asociados con una instalación de procesamiento de información, deben ser suspendidos o retirados. Esto determinará si es necesario eliminar los derechos de acceso. Los cambios de empleo deben reflejarse en la eliminación de todos aquellos derechos de acceso que no fueron aprobados para el nuevo empleo. Todas estas medidas incluyen a los derechos de acceso lógico y físico.

La eliminación o modificación, según la (ISO/IEC, 2013), se puede hacer mediante el retiro, anulación o sustitución de llaves, tarjetas de identificación, instalaciones de procesamiento de información o suscripciones. Cualquier documentación que identifica los derechos de acceso de los empleados y contratistas debe reflejar la supresión o modificación de los derechos de acceso. Si un empleado saliente o un usuario externo conocen las contraseñas de otros ID de usuarios activos, éstas se deben cambiarse al término o cambio de empleo, acuerdo o contrato de estas personas.

La (ISO/IEC, 2013) también recomienda que los derechos de acceso para la información y los activos asociados a las instalaciones de

procesamiento de información deben ser reducidos o eliminados antes de la expiración o cambio del empleo, dependiendo de la evaluación de factores de riesgo tales como:

a. Si la terminación o el cambio se inicia a petición del empleado, el usuario externo o por el área de recursos humanos, y la razón del cese.
b. Las responsabilidades actuales del empleado, usuario externo o cualquier otro usuario.
c. El valor de los activos actualmente accesibles.

Información adicional

En determinadas circunstancias, la (ISO/IEC, 2013) explica que, los derechos de acceso pueden ser asignados a un grupo de personas. En tales circunstancias, los individuos cesantes deben ser retirados de cualquier lista de acceso de grupo, y deben tomarse medidas para avisar a todos los demás empleados y usuarios externos involucrados, que ya no deben compartir información con la persona en cese.

Se observa también en la (ISO/IEC, 2013) que, en los casos de cese iniciados por el área de recursos humanos, algunos empleados descontentos o usuarios externos pueden sabotear o corromper deliberadamente la información o las instalaciones de procesamiento de información. En los casos de renuncia o despido, estas personas pueden tener la tentación de llevarse información de la empresa para su uso futuro.

2.1.2.2.5. Dominio Seguridad en las Telecomunicaciones

a) Gestión de seguridad de red (Control 13.1)

Objetivo: Asegurar la protección de la información en las redes y sus instalaciones de apoyo al procesamiento de información.

1. Controles de red (Control 13.1.1)

Control

Se debe gestionar y controlar las redes para proteger la información en los sistemas y aplicaciones.

Guía de Implementación

La recomendación de la (ISO/IEC, 2013) en esta guía indica que, se deben implementar controles para garantizar la seguridad de la información en las redes y para proteger a los servicios conectados de accesos no autorizados. De forma particular, se deben considerar los siguientes puntos:

a. Se deben establecer responsabilidades y procedimientos para la gestión de equipos de red.
b. La responsabilidad operativa de las redes y las computadoras debe ser separada.
c. Se deben establecer controles especiales para garantizar la confidencialidad e integridad de los datos que pasan a través de redes públicas o sobre redes inalámbricas y para proteger las aplicaciones y los sistemas conectados; pueden ser necesarios algunos controles especiales para mantener la disponibilidad de los servicios de red y equipos conectados.
d. Debe aplicarse un adecuado registro y seguimiento de actividades para permitir la grabación y detección de acciones que son relevantes o que pueden afectar a la seguridad de la información.
e. Se debe coordinar minuciosamente las actividades de gestión tanto para optimizar el servicio para la organización como para garantizar que los controles se aplican regularmente a través de la infraestructura de procesamiento de la información.
f. Los sistemas de red deben ser autenticados.
g. La conexión de los sistemas a la red se debe restringir.

Información adicional

Información adicional sobre seguridad de red puede encontrarse en la ISO/IEC 27033.

2. Seguridad de servicios de red (Control 13.1.2)

Control

La (ISO/IEC, 2013) recomienda que, se debe identificar e incluir en los acuerdos de servicios de red, los mecanismos de seguridad, niveles de

servicio y requisitos de gestión de todos los servicios de la red, ya sean servicios prestados en la empresa o subcontratados.

Guía de Implementación

Se debe determinar y supervisar periódicamente, según la (ISO/IEC, 2013), la capacidad que tiene el proveedor del servicio de red, para gestionar los servicios acordados de forma segura, y debe acordarse también el derecho a auditarlos.

Asimismo, la (ISO/IEC, 2013) sostiene que, se deben identificar las medidas de seguridad necesarias para servicios puntuales, tales como los aspectos de seguridad, niveles de servicio y requisitos de gestión. La organización debe asegurarse de que los proveedores de servicios de red implementen estas medidas.

Información adicional

Como información adicional, la (ISO/IEC, 2013) explica que, en los servicios de red se incluye el suministro de conexiones, servicios de redes privadas y redes de valor agregado así como soluciones de seguridad de red gestionados, por ejemplo los firewalls o los sistemas de detección de intrusos. Estos servicios pueden ir desde un simple ancho de banda no administrado hasta complejas ofertas de valor agregado.

Las características de seguridad aplicadas a estos servicios de red podrían ser:

a. La tecnología utilizada para la seguridad de los servicios de red, como los controles de conexión, autenticación y encriptación.
b. Los parámetros técnicos necesarios para tener conexiones seguras con los servicios de red de acuerdo con las reglas de seguridad y conexión a la red.
c. De ser necesario, tener procedimientos de uso de servicios de red para restringir el acceso a los servicios de red o aplicaciones.

2.1.2.3. Dimensión metodológica

El autor ha creído necesario establecer una guía metodológica propia que sirva de base para la operacionalización de la variable dependiente, y que contribuya a una mejor comprensión del problema y sus consecuencias directas antes y después de diseñado el protocolo.

El autor para esto presenta un enfoque de una dimensión metodológica denominada *evaluación*. Esta propuesta permite una medición más precisa sobre la revisión de los derechos de acceso de los usuarios.

2.1.2.3.1. Evaluación

Es una dimensión metodológica que consiste en señalar el valor actual y final de 2 indicadores muy importantes en la revisión de los derechos de acceso de los usuarios: la performance y la seguridad.

La performance es el indicador del desempeño humano medido en minutos que, una persona familiarizada con la administración de redes y sistemas operativos, invierte para completar las actividades relacionadas con la revisión de información sobre los usuarios. Estas acciones están basadas en los puntos de control 9.2.5 de la ISO/IEC 27002:2013 y su medición está sujeta a la experiencia práctica del personal a cargo de los servidores pero validada en un entorno de producción real.

Las actividades relacionadas con los puntos de control mencionados deben ser medidas de forma individual en segundos, sin embargo el tiempo acumulado por todas ellas debe estar indicado en minutos. Entre las actividades más importantes se encuentran:

- ✓ Login o inicio de sesión al sistema operativo
- ✓ Observar información del usuario
- ✓ Observar información de privilegios
- ✓ Observar información sobre el grupo del usuario
- ✓ Crear un nuevo usuario o grupo
- ✓ Modificar un usuario o grupo
- ✓ Eliminar un usuario o grupo
- ✓ Validar la información observada

El investigador puede incluir o retirar alguna de estas actividades durante su proyecto según sea su criterio personal.
La categorización para este indicador está dada por los siguientes rangos de valores:

Tabla 14
Cuadro de categorización de performance para dimensión metodológica de Evaluación

Xi	Rango de tiempo reducido	Categoría
1	0-1: De cero a 1 minuto incluido 1:59	Optimo
3	2-3: De 2 a 3 minutos incluido 3:59	Aceptable
5	4-9: De 4 a 9 minutos incluido 9:59	Medio
10	10 a más: De 10 a más minutos	Bajo

Fuente: Elaboración del investigador

Los rangos de tiempo se reducen a valores decimales enteros (redondeados) para una evaluación estadística mucho más sencilla. Aunque el uso de segundos en lugar de minutos puede considerarse, el autor ha creído conveniente utilizar los minutos para la evaluación final por ser más sencilla de comprender para cualquier persona que utilice el sistema decimal de unidades.

La seguridad es el indicador que mide el grado de fiabilidad en la transmisión de los mensajes generados por el protocolo diseñado, enfocándose en la integridad y la confidencialidad. La disponibilidad no está considerada en este indicador debido a que no forma parte del diseño de un protocolo sino más bien que responde a políticas técnicas sobre la entrega del servicio. Este indicador está basado en los puntos de control 13.1.2 de la ISO/IEC 27002:2013 y su medición también está sujeta a la experiencia práctica del personal a cargo de los servidores y validada en un entorno de producción real.

Los aspectos relativos a la seguridad que se evalúan son 3: la autenticación, el control de conexiones y la encriptación. A partir de la evaluación individual se procede con la categorización de la seguridad según el siguiente cuadro.

Tabla 15

Cuadro de categorización de seguridad para dimensión metodológica de Evaluación

Autenticación	**Control de Conexiones**	**Encriptación (tamaño de clave)**	**Categoría**
NO	NO	NO	Crítico
SI	NO	SI (<= 512bits)	Bajo
SI	NO	SI (>= 1024bits y <= 2048bits)	Medio
SI	SI	SI (>= 1024bits y <= 2048bits)	Aceptable
SI	SI	SI (>= 4096bits)	Optimo

Fuente: Elaboración del investigador

La autenticación permite conocer si se utilizan mecanismos de verificación para acceder al servidor que brinda el sistema o servicio. El control de conexiones por su parte permite saber si se usan mecanismos de control de red para que solo acceda personal autorizado al servidor que brinda el sistema o servicio (p.e. firewall rules, acl, etc.). Con la autenticación y el control de conexiones se cubre la confidencialidad en la seguridad de la información. La integridad en cambio, se identifica por el nivel de encriptación (determinado por el número de bits para la clave) utilizado para la transmisión de información por la red en el servidor que brinda el sistema o servicio y es medido en bits que van desde cero (sin nivel de encriptación) hasta más de 1024 bits que es considerado actualmente según (Lucas, 2012) un nivel aceptable de encriptación. Un número mayor a 4096 por consiguiente es considerado en esta investigación como un nivel óptimo.

Finalmente, la evaluación final sobre la seguridad se da en base a las categorías, siendo de estas la más baja la categoría “Crítico” que denota una completa ausencia de la seguridad y la categoría “Optimo” que es la más recomendada para cualquier sistema operativo.

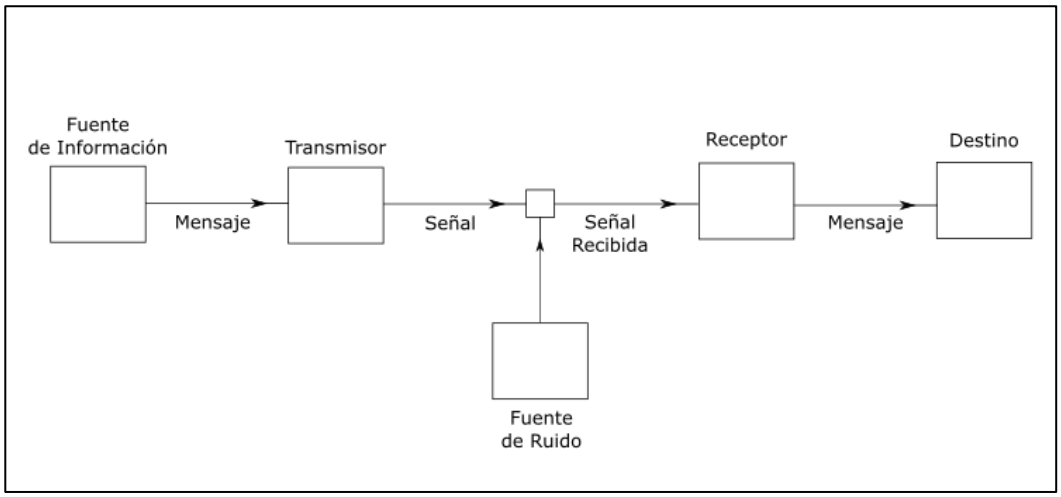
2.2. Teoría científica que sustentan las variables del estudio

2.2.1. Teoría matemática de la comunicación

Los investigadores Claude E. Shannon y Warren Weaver en su libro “The Mathematical Theory of Communication” abordan la teoría de la comunicación como parte específica (campo) de la teoría de la información.

El propósito de toda comunicación, según (Shannon & Weaver, 1963), es influenciar en la conducta del receptor por parte del transmisor, y que para conseguir esto, la comunicación se enfrenta a diferentes problemas en tres niveles (Nivel A, B y C). El nivel A, denominado *problema técnico*, se pregunta cómo pueden ser transmitidos de forma fiable los símbolos de la comunicación. El nivel B, denominado *problema semántico*, pregunta cómo los simbolos transmitidos entregan el significado deseado. Y el nivel C, se pregunta que tan efectivo resulta el mensaje recibido en el comportamiento del receptor.
(Shannon & Weaver, 1963) a su vez, consideran que un sistema general de comunicación se compone de los siguientes elementos:

Gráfico 8: El sistema de comunicación



Fuente: (Shannon & Weaver, 1963, p. 34)

Estos elementos están presentes en cualquier tipo de comunicación

- ✓ **Fuente de información** (lo que produce el mensaje)
- ✓ **Transmisor** (aquel que cambia el mensaje en señales y lo envía al receptor a través del canal de comunicación)
- ✓ **Mensaje** (palabras, imágenes, sonidos, etc.)

- ✓ **Señal** (mensaje que ha sido transformado del tal forma que pueda viajar por el canal de comunicación)
- ✓ **Señal recibida** (señal alterada o no, que el receptor obtiene del canal de transmisión)
- ✓ **Canal de comunicación** (medio utilizado para transmitir las señales desde el transmisor al receptor)
- ✓ **Receptor** (aquel que reconstruye el mensaje, realizando la operación inversa hecha por el transmisor)
- ✓ **Destino** (la persona o cosa para quien el mensaje fue enviado)
- ✓ **Fuente de ruido** (alteración no deseada de la señal ocurrida en el canal durante la transmisión)

Esta teoría es la base de todo protocolo, incluyendo al protocolo SUMP, debido a que presenta los elementos esenciales de toda comunicación que se debe tener en cuenta durante el diseño de un protocolo.

2.2.2. Texto sobre Ingeniería de Protocolos

El profesor Hartmut König del Departamento de Ciencias de la Computación en la Brandenburg University of Technology Cottbus – Alemania en su texto "Protocol Engineering" establece los principios que rigen los protocolos de comunicación, su descripción, su diseño y su desarrollo.

En su investigación, (König, 2012) aborda los fundamentos de los protocolos de comunicación introduciendo conceptos y mecanismos básicos de protocolos. Así mismo, afirma el origen de los protocolos en los paradigmas de la comunicación humana de la vida diaria. Simultáneamente introduce fundamentos importantes para describir la comunicación de servicios y protocolos. Los principios de protocolos que destaca son:

- ✓ Servicios
- ✓ Protocolos
- ✓ Capas
- ✓ Arquitectura de capas

En su libro también desarrolla un protocolo denominado XDT (eXample Data Transfer) el cual aborda con una metodología propuesta pero opcional para la

descripción de la funcionalidad como el desarrollo y validación del protocolo. Estos elementos han servido como guía para el diseño del protocolo SUMP.

2.2.3. Norma ISO/IEC 27002:2013

Este estándar internacional está diseñado por organizaciones para ser usado como referencia en la selección de controles al momento de implementar un SGSI (Sistema de Gestión de Seguridad de la Información) basado en la ISO/IEC 27001, o como un documento guía para organizaciones que se encuentren implementando controles de seguridad mundialmente aceptados.

La norma *ISO/IEC 27002:2013 Information technology - Security techniques - Code of practice for information security controls* proporciona entre otras guías, una dedicada a la gestión de usuarios y el tratamiento de la seguridad. Estos puntos resultan ser cruciales para solucionar el problema abordado en la presente tesis.

Según la (ISO/IEC, 2013), la ISO (the International Organization for Standardization) y la IEC (the International Electrotechnical Commission) forman el sistema especializado de estandarización a nivel mundial. Organismos nacionales que son miembros de la ISO o IEC participan en el desarrollo de estándares internacionales a través de comités técnicos establecidos por la respectiva organización para lidiar con campos particulares de actividad técnica.

La Norma ISO/IEC 27002:2013 proporciona un sustento científico sólido para el tratamiento de la revisión de los derechos de acceso de los usuarios. Además brinda un marco mucho más amplio que complementa con creces el ámbito de esta investigación.

2.3. Marco Conceptual

Según (Behar, 2008), el marco conceptual deriva una serie de postulados que expresan relaciones entre las variables estudiadas de forma deductiva. Para tal efecto, se muestran los siguientes postulados.

2.3.1. Protocolo

Conjunto de reglas que se definen previamente para permitir la comunicación entre computadoras con el objetivo de transmitir información relevante a un propósito determinado

2.3.2. Cuenta de Usuario

Identificador único para una persona o entidad que se encuentra registrado en algún computador o servicio con la finalidad de poder hacer uso de sus recursos.

2.3.3. Privilegio

Permiso específico que se le asigna a una o varias cuentas de usuario con el objetivo de restringir u otorgar acceso a los recursos que se consideren convenientes.

2.3.4. MySQL

Es un sistema gestor de base de datos open source ampliamente utilizado en aplicaciones web. Permite utilizar el lenguaje estándar SQL para manipular la información de base de datos, tablas y relaciones.

2.3.5. Linux

Es un sistema operativo libre, basado en Unix. Es uno de los principales ejemplos de software libre y de código abierto. Linux está licenciado bajo la GPL v2 y está desarrollado por colaboradores de todo el mundo.

CAPITULO III
MARCO METODOLOGICO

3. CAPITULO III – MARCO METODOLOGICO

3.1. Hipótesis

Hipótesis de investigación (Hi)

Si se aplica el protocolo SUMP, entonces se mejora significativamente la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en la sede Iquitos.

Hipótesis nula (Ho)

Si se aplica el protocolo SUMP, entonces no se mejora significativamente la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en la sede Iquitos.

3.2. Definición de Variables

3.2.1. Definición Conceptual

a) V.I: Protocolo SUMP

Un protocolo (protocolo de comunicación), tal como lo explica (Sharp, 2008), es un conjunto de reglas para intercambiar determinados tipos de información ya sea contenido general (señales electrónicas o de gran cantidad de información) o del tipo de codificado que implican también un conjunto de reglas para codificar varios tipos de mensajes. Este intercambio de datos puede tomar lugar entre varias partes. Cuando hay N partes para su intercambio debemos hablar de una comunicación N-peer y referirnos al protocolo como un protocolo N-peer.

Así mismo, (Sharp, 2008) también sostiene que por cada una de las N partes en una comunicación N-Peer, el protocolo define un lenguaje, cuyas sentencias son secuencias legales de mensajes recibidos por esa parte, y cuyo alfabeto de símbolos son el conjunto de todos los posibles mensajes. Para que una maquina obedezca las reglas del protocolo debe ser capaz de reconocer el lenguaje del protocolo.

El protocolo SUMP (Simple User Management Protocol), es una especificación de un protocolo diseñado para intercambiar información sobre cuentas de usuarios y privilegios siguiendo las recomendaciones de la norma ISO/IEC 27002:2013 en su control 9.2.5

b) V.D: Revisión de los derechos de acceso de los usuarios

La revisión de los derechos de acceso de los usuarios (control 9.2.5), según sostiene la (ISO/IEC, 2013), forma parte de un serie de guías de las mejores prácticas realizadas por las empresas e instituciones en materia de seguridad de la información. Concretamente está desarrollada bajo el dominio de Control de Accesos y del objetivo de control para la Gestión de Accesos de usuario.

La norma establece una guía de implementación basada en la revisión periódica de las cuentas de usuarios y sus derechos de acceso así como el monitoreo de cuentas privilegiadas.

3.2.2. Definición Operacional

a) V.I: Protocolo SUMP

Para la presente investigación se usara la metodología de diseño mixta propuesta por el autor como principal guía en diseño de un protocolo y se utilizará el lenguaje Go como lenguaje de programación para implementar una simulación del protocolo SUMP en un entorno virtualizado con servidores Red-Hat bajo la plataforma Open VZ.

b) V.D: Revisión de los derechos de acceso de los usuarios

El investigador ha considerado trabajar con el control 9.2.5 de la norma ISO/IEC 27002:2013 como punto de partida para la traducción de las guías de seguridad propuestas para ser incorporada como funcionalidad en el protocolo de red.

3.2.3. Operacionalización de Variables

Tabla 16
Operacionalización de variables

Variables	Dimensiones	Indicadores	Categoría
V.I: Protocolo SUMP	Selección	❖ Obtener población de servidores	Completo Incompleto
		❖ Identificar sistemas y servicios	Requiere Monitoreo (SI/NO)
		❖ Elaborar muestra de trabajo	Prioridad Alta Prioridad Media

			Prioridad Baja
	Diseño	❖ Elaborar Propuesta de Protocolo	Optimo Aceptable Deficiente
V.D: Revisión de los derechos de acceso de los usuarios	Evaluación	❖ Performance	0-1: Optimo 2-3: Aceptable 4-9: Medio 10 a más: Bajo
		❖ Seguridad	Crítico: Sin Autenticación, Sin Control de Conexiones, sin encriptación Bajo: Con Autenticación, Sin Control de Conexiones, Con encriptación <=512bits Medio: Con Autenticación, Sin Control de Conexiones, Con encriptación >= 1024bits y <= 2048bits Aceptable: Con Autenticación, Con Control de Conexiones, Con encriptación >= 1024bits y <= 2048bits Optimo: Con Autenticación, Con Control de Conexiones, Con encriptación >=4096bits

Fuente: Elaboración del investigador

3.3. Metodología

3.3.1. Tipo de Estudio o Investigación

La presente investigación es de tipo **aplicada** ya que este tipo de investigación “es el estudio y aplicación de la investigación a problemas concretos, en circunstancias y características concretas. Esta forma de investigación se dirige a su aplicación inmediata y no al desarrollo de teorías” (Behar, 2008, p. 20). Se busca pues, resultados inmediatos que permitirán demostrar la eficacia del diseño del protocolo SUMP.

Así mismo, esta investigación también es de tipo **experimental** porque “el investigador actúa conscientemente sobre el objeto de estudio, en tanto que los objetivos de estos estudios son precisamente conocer los efectos de los actos producidos por el propio investigador como mecanismo o técnica para probar sus hipótesis” (Bernal, 2010, p. 117). Mediante la manipulación a la variable independiente, esta investigación busca probar la hipótesis.

3.3.2. Diseño de Investigación

El diseño de investigación elegido por el investigador ha sido el diseño **Pre-experimental**, donde “a un grupo se le aplica una prueba previa al estímulo o tratamiento experimental, después se le administra el tratamiento y finalmente se le aplica una prueba posterior al estímulo.” (Hernández, 2014, p. 141).

El diagrama elegido es el pre-test/post-test con un solo grupo

GE: O1 ---------- X ---------- O2

Dónde:

GE = Grupo experimental
O1 = Pre-test
X = Protocolo SUMP
O2 = Post-test

3.4. Población y Muestra

3.4.1. Población

La población “es el conjunto de individuos que tienen ciertas características o propiedades que son las que se desea estudiar” (Fuentelsaz, Pulpón, & Icard, 2006, p. 55). Y para el presente trabajo de investigación, la población está conformada por todos los servidores en funcionamiento que son utilizados en la empresa Petroperú en su sede Selva (Iquitos) y sus propiedades de estudio son las siguientes.

Tabla 17
Población. Nro. de servidores en la empresa Petroperú-Iquitos

Servidor	**Sistema Operativo**	**Sistema o Servicio**	**Descripción**	**Porcentaje**
Proxy Server	RHEL	Linux	Sistema RHEL para servidor squid	3.125%
Proxy Server	RHEL	Squid	Servicio Squid	3.125%
Proxy Server	RHEL	HTTP	Servicio HTTP	3.125%
Mail Server	RHEL	Linux	Sistema RHEL para servidor de correo	3.125%
Mail Server	RHEL	SMTP	Servicio SMTP	3.125%
Mail Server	RHEL	HTTP	Servicio de soporte HTTP para el servicio SMTP	3.125%
SMB Server	RHEL	Linux	Sistema RHEL para servidor de archivos	3.125%
SMB Server	RHEL	Samba	Servicio SMB	3.125%
WSUS Server	Windows	WSUS	Servicio de actualización para equipos windows	3.125%
Central Server	RHEL	Linux	Sistema RHEL para servidor central	3.125%
Central Server	RHEL	MySQL	Servicio de base de datos MySQL	3.125%
Central Server	RHEL	HTTP	Servicio HTTP	3.125%
Central Server	RHEL	DB2	Servicio de base de	3.125%

			datos DB2	
Backup Server	RHEL	Linux	Sistema RHEL para servidor de Backups	3.125%
Backup Server	RHEL	DB2	Servicio de base de datos DB2	3.125%
Virt Server 1	VMware	ESX	Sistema VMware para virtualización de servidores	3.125%
Syslog Server	RHEL	Linux	Sistema RHEL para servidor Syslog	3.125%
Monitor Server 1	RHEL	Linux	Sistema RHEL para servidor de monitoreo 01	3.125%
Monitor Server 1	RHEL	MySQL	Servicio de base de datos MySQL	3.125%
Monitor Server 1	RHEL	Ntop	Servicio de monitoreo de ancho de banda ntop	3.125%
Intranet	Windows	Windows	Sistema Windows para el servidor intranet	3.125%
Intranet	Windows	Terminal Services	Servicio de acceso remoto	3.125%
Monitor Server 2	RHEL	Linux	Sistema RHEL para servidor de monitoreo 02	3.125%
Monitor Server 2	RHEL	Ntop	Servicio de monitoreo de ancho de banda ntop	3.125%
Monitor Server 2	RHEL	MySQL	Servicio de base de datos MySQL	3.125%
App Server	Windows	Windows	Sistema Windows para el servidor de aplicaciones	3.125%
Virt Server 2	VMware	ESX	Sistema VMware de Backup para virtualización de servidores	3.125%
Directory	RHEL	Linux	Sistema Linux RHEL	3.125%

Server			para servidor de directorio	
Directory Server	RHEL	LDAP	Servicio de directorio	3.125%
Directory Server	RHEL	DB2	Servicio de base de datos DB2	3.125%
Directory Server	RHEL	DHCP	Servicio DHCP	3.125%
Directory Server	RHEL	DNS	Servicio DNS	3.125%
			Total (32)	**100%**

Fuente: Relación de servidores activos según contrato ML.2014

Fecha: 2014-12-14

3.4.2. Muestra

La muestra es “el grupo de individuos que realmente se estudiarán, es un subconjunto de la población” (Fuentelsaz, Pulpón, & Icard, 2006, p. 55). Asi mismo, la técnica utilizada para obtener la muestra en la presente investigación ha sido determinada por muestreo intencional que consiste en que “todos los elementos muestrales de la población serán seleccionados bajo estricto juicio personal del investigador” (Naghi Namakforoosh, 2014, p. 189).

Tabla 18
Muestra. Nro. de sistemas y servicios en la empresa Petroperú-Iquitos

Servidor	**Sistema o Servicio**	**Descripción**	**Porcentaje**
Proxy Server	Linux	Sistema RHEL para servidor squid	9.091%
Mail Server	Linux	Sistema RHEL para servidor de correo	9.091%
SMB Server	Linux	Sistema RHEL para servidor de archivos	9.091%
Central Server	Linux	Sistema RHEL para servidor central	9.091%
Central Server	MySQL	Servicio de Base de Datos MySQL	9.091%
Backup Server	Linux	Sistema RHEL para servidor de	9.091%

		Backups	
Syslog Server	Linux	Sistema RHEL para servidor Syslog	9.091%
Monitor Server 1	Linux	Sistema RHEL para servidor de Monitoreo 1	9.091%
Monitor Server 1	MySQL	Servicio de Base de Datos MySQL	9.091%
Monitor Server 2	MySQL	Subsistema de Base de Datos MySQL	9.091%
Directory Server	Linux	Sistema RHEL para servidor de Directorio	9.091%
		Total (11)	**100%**

Fuente: Tabla 17

Fecha: 2014-12-14

3.5. Métodos de Investigación

En la presente investigación se ha trabajado con los siguientes métodos de investigación generales.

Método Inductivo: "es un proceso en el que, a partir del estudio de casos particulares, se obtienen conclusiones o leyes universales que explican o relacionan los fenómenos estudiados". (Rodríguez, 2005, p. 29). Este método fue utilizado en la observación directa del problema, la experimentación mediante la aplicación del estímulo y el estudio de la relación existente entre ellos.

Método Deductivo: “Consiste en obtener conclusiones particulares a partir de una ley universal”. (Rodríguez, 2005, p. 29). Este método fue usado para formular la hipótesis de la investigación así como la deducción de las conclusiones.

Método Analítico: “En este método se distinguen los elementos de un fenómeno y se procede a revisar ordenadamente cada uno de ellos por separado.” (Rodríguez, 2005, p. 30). Con este método se pudo analizar la información obtenida en las diferentes etapas del proyecto de investigación.

Método Sintético: “Es un proceso mediante el cual se relacionan hechos aparentemente aislados y se formula una teoría que unifica los diversos

elmentos". (Rodríguez, 2005, p. 30). Este método fue usado para integrar el conocimiento de la investigación que permitio elaborar el marco teórico y conceptual.

3.6. Técnicas e Instrumentos de recolección de datos

3.6.1. Técnicas de recolección de datos

"Las técnicas de recolección de datos comprenden procedimientos y actividades que le permiten al investigador obtener la información necesaria para dar respuesta a su pregunta de investigación". (Hurtado de Barrera, 2000, p. 427)

La técnica utilizada en el presente proyecto fue:

3.6.1.1. Encuesta

El investigador ha utilizado esta técnica para recolectar información que refleje la situación actual del problema en relación al tratamiento de las cuentas de usuarios en la empresa Petroperú sede Iquitos.

3.6.2. Instrumento de recolección de datos

Los instrumentos de recolección de datos "constituyen las vías mediante la cual es posible aplicar una determinada técnica de recolección de información". (Hurtado de Barrera, 2000, p. 427)

Los instrumentos utilizados en el presente proyecto fueron:

3.6.2.1. Cuestionario

El investigador ha utilizado este instrumento para recolectar información sobre la lista de servidores y servicios que requieren de un monitoreo y control de las cuentas de usuarios y privilegios.

3.6.2.2. Pre-Test

El investigador ha utilizado este instrumento para recolectar información sobre el estado actual del proceso de gestión de cuentas de usuario.

3.6.2.3. Post-Test

El investigador ha utilizado este instrumento para recolectar información sobre el estado del proceso de gestión de cuentas de usuario durante la simulación del protocolo SUMP.

3.7. Métodos de análisis de datos

En la presente investigación se ha utilizado Microsoft Excel y SPSS.

El análisis de información se realizó usando el análisis cuantitativo mediante el trabajo estadístico. Así mismo se tuvo en cuenta cuadros estadísticos para exponer los datos que se obtuvieron al aplicar los instrumentos de recolección, y la posterior aplicación de los siguientes estadígrafos:

3.7.1. Medida de Tendencia Central

Media aritmética de datos agrupados ($\bar{x}$): Valor obtenido al agrupar n valores iguales, cada uno por separado, considerando su frecuencia. Se usará para la obtención del promedio de los datos de la muestra.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i f_i}{n}$$

Dónde:

$\bar{x}$ = Media aritmética.

$\sum$ = Sumatoria.

f_i = Frecuencia.

x_i = Valores observados

n = Muestra

3.7.2. Medidas de Dispersión

Desviación Estándar para datos agrupados (S): La desviación estándar nos da como resultado un valor numérico que representa el promedio de diferencia que hay entre los datos y la media.

$$S = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2 \, f_i}{n-1}}$$

Dónde:

S = Desviación estándar

$\sum$ = Sumatoria

f_i = Frecuencia absoluta

x_i = Valores observados

$\bar{x}$ = Media aritmética

n = Muestra

Coeficiente de variación (Cv): Permite comparar la variación producto de 2 variables diferentes que pueden provenir de una misma muestra

$$C_V = \frac{s}{\bar{x}}(100)$$

Dónde:

C_v = Coeficiente de variación

S = desviación estándar

$\bar{x}$ = Media aritmética

100 = Valor porcentual constante

CAPITULO IV
RESULTADOS DE LA INVESTIGACION

4. CAPITULO IV – RESULTADOS DE LA INVESTIGACION

4.1. Análisis e Interpretación de los resultados

4.1.1. Objetivo 1

Identificar los diferentes sistemas y servicios utilizados en los servidores de Petroperú en su sede Iquitos que requieren de un protocolo de gestión de usuarios

Para cumplir este objetivo se realizaron las siguientes actividades

Tabla 19
Actividades para cumplir el objetivo 1

Actividad	Descripción
Obtener población de servidores	Recopilar información sobre la población de servidores utilizados en Petroperú-Iquitos.
Identificar sistemas y servicios	Identificar los sistemas y servicios de cada servidor con sistema operativo RHEL que utilicen cuentas de usuarios y privilegios entregados a los empleados como parte de su servicio de infraestructura tecnológica
Elaborar muestra de trabajo	Elaborar un cuadro con todos los sistemas y servicios utilizados en Petroperú-Iquitos que requieran de un monitoreo de las cuentas de usuarios y sus privilegios de tal forma que pueda concluir el objetivo propuesto

Fuente: Elaboración del investigador

a) Obtener población de servidores

Para el desarrollo de esta actividad se solicitó el apoyo al Jefe de la Unidad de TI y Comunicaciones de Petroperú, el Ing. Tony Vásquez Vásquez, quien derivó al Ing. Marks Navarro Yuyarima, la entrega del documento con la relación de servidores según contrato de outsourcing con IBM 2014.Esta información proporcionada fue vital para continuar con la investigación.

A la actividad "Obtener población de servidores", se le dio la categoría de COMPLETO, debido a que el personal de Petroperú sede Iquitos brindo oportunamente toda la información solicitada que fue necesaria para esta investigación (ver Anexo 14 – “Documento de requerimiento de información de servidores en la EMPRESA PETROPERÚ sede Iquitos”).

b) Identificar sistemas y servicios

Esta actividad se llevó a cabo con el apoyo del Supervisor de Infraestructura de la Unidad de TI y Comunicaciones de Petroperú, el Ing. Marks Navarro Yuyarima, quien completó el Cuestionario de Identificación de Sistemas y Servicios (Ver Anexo 01). Se han considerado las siguientes preguntas técnicas para la evaluación de los sistemas y servicios: ¿Usa Cuentas de Usuarios Locales? y ¿Usa Privilegios Locales?, cumpliendo con la Estructura de cuestionario de identificación de sistemas y servicios. Estas preguntas técnicas fueron propuestas debido a la relación directa que tienen con el propósito del protocolo a diseñar.

Tabla 20
Sistemas y Servicios de Petroperú-Iquitos

Servidor	**Sistema / Servicio**	**Usa Cuentas de Usuarios Locales**	**Usa Privilegios Locales**	**Requiere Monitoreo y Control**
Proxy Server	Linux	SI	SI	SI
Proxy Server	Squid	NO	NO	NO
Proxy Server	HTTP	NO	NO	NO
Mail Server	Linux	SI	SI	SI
Mail Server	SMTP	NO	NO	NO
Mail Server	HTTP	NO	NO	NO
SMB Server	Linux	SI	SI	SI
SMB Server	Samba	SI	SI	SI
Central Server	Linux	SI	SI	SI
Central Server	MySQL	SI	SI	SI
Central Server	HTTP	SI	SI	SI
Central Server	DB2	SI	SI	NO
Central Server	Squid	NO	NO	NO
Backup Server	Linux	SI	SI	SI
Backup Server	HTTP	NO	NO	NO
Backup Server	DB2	SI	SI	SI
Backup Server	TSM	SI	SI	NO
Syslog Server	Linux	SI	SI	SI
Syslog Server	Syslog	NO	NO	NO
Monitor Server 1	Linux	SI	SI	SI
Monitor Server 1	HTTP	NO	NO	NO
Monitor	MySQL	SI	SI	SI

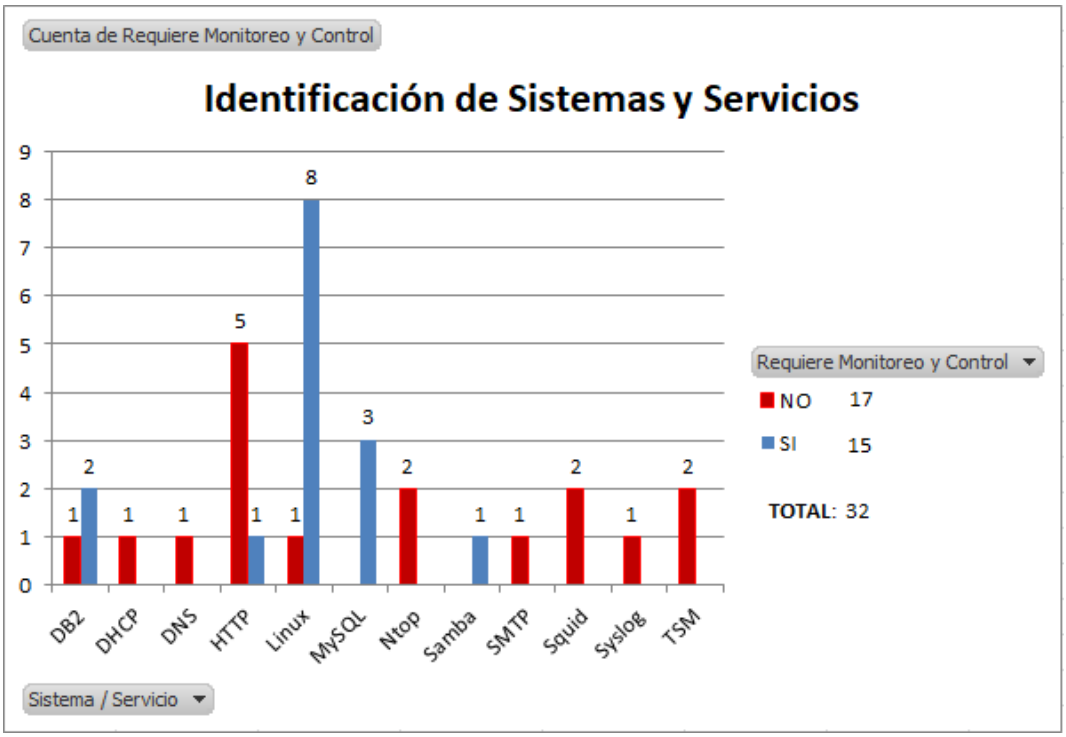
Server 1 Monitor	Ntop	SI	SI	NO
Server 1 Monitor	Linux	SI	SI	NO
Server 2 Monitor	HTTP	NO	NO	NO
Server 2 Monitor	MySQL	SI	SI	SI
Server 2 Monitor	Ntop	SI	SI	NO
Server 2 Directory	Linux	SI	SI	SI
Server Directory	TSM	SI	SI	NO
Server Directory	DB2	SI	SI	SI
Server Directory	DHCP	NO	NO	NO
Server Directory	DNS	NO	NO	NO
Server				

Fuente: Cuestionario de Identificación de Sistemas y Servicios Petroperú-Iquitos

Fecha: 2014-12-14

Esta tabla nos muestra la compilación de los resultados del cuestionario de identificación de sistemas y servicios aplicado en Petroperú sede Iquitos, de la cual podemos concluir que se encontraron 32 servidores Linux, de los cuales 21 sistemas y servicios sí utilizan cuentas de usuarios locales; por otro lado, se puede apreciar que 21 servidores si utilizan privilegios locales. Así mismo se observa que 15 sistemas y servicios requieren monitoreo y control.

Gráfico 9: Identificación de Sistemas y Servicios



Fuente: Tabla 20

Fecha: 2014-12-14

Este gráfico nos muestra que se identificaron 32 sistemas y servicios Linux que están agrupados en: DB2(3), DHCP(1), DNS(1), HTTP(6), Linux(9), MySQL(3), Ntop(2), Samba(1), SMTP(1), Squid(2), Syslog(1), TSM(2). Estos sistemas y servicios se categorizaron de la siguiente manera: 17 de ellos no requieren monitoreo y control, esto quiere decir que para Petroperú, es innecesario obtener información de un monitoreo de cuentas de usuario para: DB2(1), DHCP(1), DNS(1), HTTP(5), Linux(1), NTOP(2), SMTP(1), Squid(2), Syslog(1), TSM(2).

Por otro lado, 15 sistemas y servicios si requieren monitoreo y control, esto quiere decir que es deseable un monitoreo de cuentas de usuario para: DB2(2), HTTP(1), Linux(8), MySQL(3), Samba(1).

c) Elaborar muestra de trabajo

Esta actividad se llevó a cabo con el apoyo del Supervisor de Infraestructura UTIC el Ing. Marks Navarro Yuyarima, quien completó el Cuestionario de priorización de sistemas y servicios (Ver Anexo 02), en base a la información recolectada en el paso b; es decir, después de obtener solamente aquellos servicios que requieren monitoreo y control, se procede a evaluar su prioridad.

Esta muestra de trabajo representa un filtro más para delimitar el alcance que tiene el protocolo, y se utiliza la categorización por 3 prioridades (Alta, Media y Baja) según la metodología propuesta por el investigador.

Tabla 21
Priorización de Sistemas y Servicios de Petroperú-Iquitos

Servidor	**Sistema o Servicio**	**Descripción**	**Prioridad**
Proxy Server	Linux	Sistema RHEL para servidor squid	Alto
Mail Server	Linux	Sistema RHEL para servidor de correo	Alto
SMB Server	Linux	Sistema RHEL para servidor de archivos	Alto
SMB Server	Samba	Servicio SMB	Medio
Central Server	Linux	Sistema RHEL para servidor central	Alto
Central Server	MySQL	Servicio de Base de Datos MySQL	Alto
Central Server	HTTP	Subsistema HTTP	Bajo
Backup Server	Linux	Sistema RHEL para servidor de Backups	Alto
Backup Server	DB2	Servicio de Base de Datos DB2	Medio
Syslog Server	Linux	Sistema RHEL para servidor Syslog	Alto
Monitor Server 1	Linux	Sistema RHEL para servidor de Monitoreo 1	Alto
Monitor Server 1	MySQL	Servicio de Base de Datos MySQL	Alto
Monitor Server 2	MySQL	Servicio de Base de Datos MySQL	Alto
Directory Server	Linux	Sistema RHEL para servidor de Directorio	Alto
Directory Server	DB2	Servicio de Base de Datos DB2	Medio

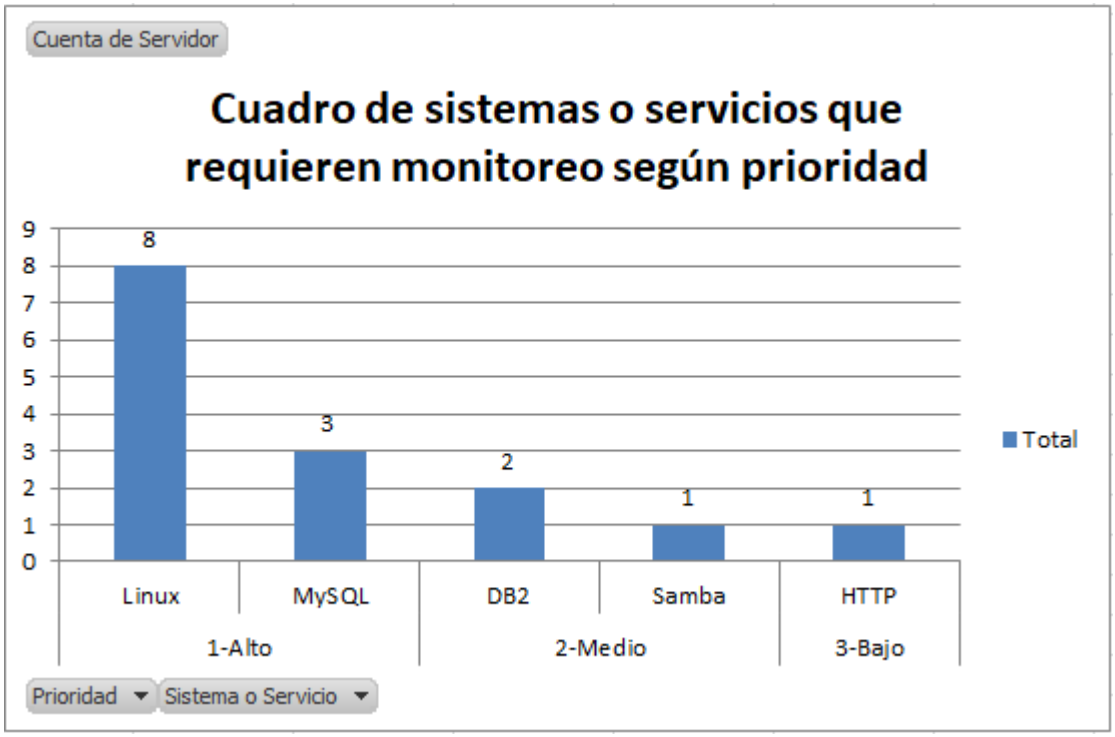
Fuente: Cuestionario de Priorización de Sistemas y Servicios Petroperú-Iquitos

Fecha: 2014-12-14

Esta tabla nos muestra que tras la aplicación del cuestionario de priorización de sistemas y servicios, se encontró que existen 11 sistemas y servicios con prioridad Alta, 3 sistemas y servicios con prioridad Media y 1 sistema y servicio con prioridad Baja.

Esta nueva categorización por prioridad permitió definir la muestra final en la que se basará el desarrollo del protocolo SUMP. Los resultados que se obtuvieron los podemos observar mejor en el siguiente gráfico:

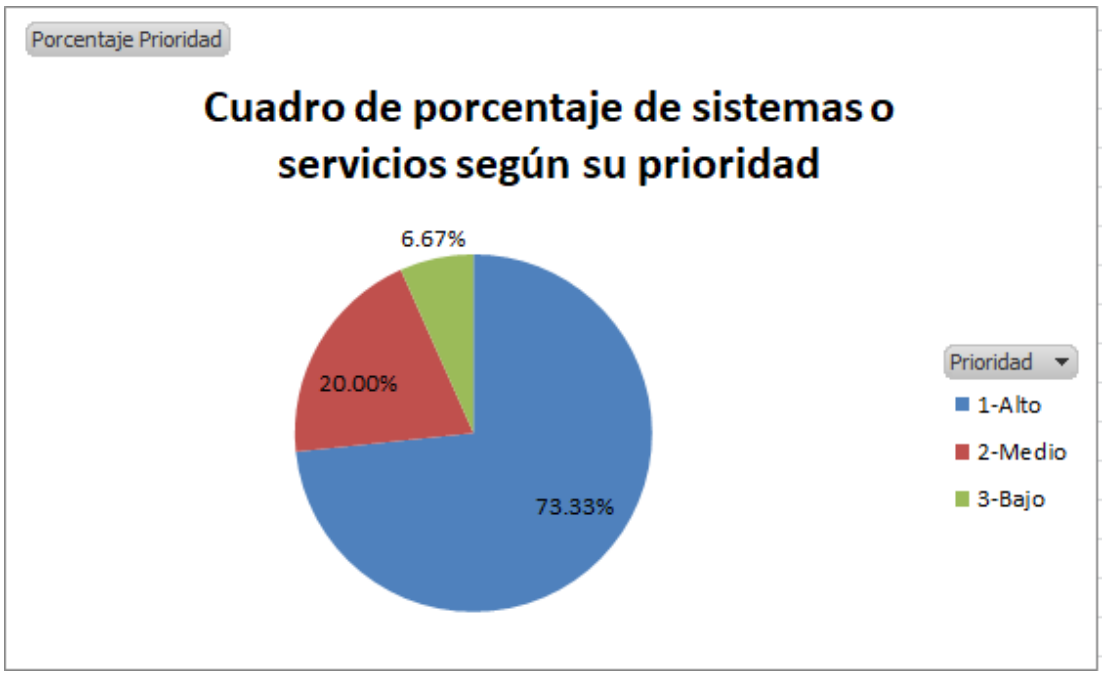
Gráfico 10: Sistemas o servicios que requieren monitoreo según prioridad



Fuente: Tabla 21

Fecha: 2014-12-14

Gráfico 11: Porcentaje de sistemas o servicios que requieren monitoreo según prioridad



Fuente: Tabla 21

Fecha: 2014-12-14

En el gráfico 10 y 11, se puede apreciar que el 73.33% de sistemas o servicios, es decir 11 de un total de 15 se ubican en la categoría Prioridad Alta; estos sistemas y servicios tienen la más alta prioridad de implementación, lo que significa que deben estar incluidos dentro de las pruebas de simulación del protocolo SUMP.

Los sistemas o servicios consideradores dentro de la prioridad alta fueron: Linux(8) y MySQL(3).

En la categoría Prioridad Media, se obtuvo que el 20.00% representado por 3 sistemas o servicios tienen una prioridad media de implementación, lo que significa que su inclusión dentro de las pruebas de simulación del protocolo SUMP es opcional y puede formar parte o no dentro de los resultados oficiales de esta investigación. Sin embargo, el investigador ha considerado conveniente prescindir de su inclusión en las pruebas de simulación.

Los sistemas o servicios considerados dentro de la prioridad media son: DB2(2) y Samba(1).

En la categoría Prioridad Baja, se obtuvo que el 6.67% representado por 1 sistema o servicio tiene una prioridad baja de implementación, lo que significa que su inclusión dentro de las pruebas de simulación del protocolo SUMP también es opcional, pero que no forma parte dentro de los resultados oficiales de esta investigación. El autor de esta investigación ha considerado conveniente prescindir de su inclusión dentro de las pruebas de simulación.

Los sistemas o servicios considerados dentro de la prioridad baja son: HTTP(1).

4.1.2. Objetivo 2

Determinar el estado actual del proceso de revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en su sede Iquitos.

Relativo a la Performance

A los sistemas y servicios que conforman el grupo de estudio se les aplicó el Pre test, con el propósito de determinar el tiempo en minutos (Xi) que toma atender requerimientos según el proceso actual de monitoreo de las cuentas de usuarios y sus privilegios y los resultados fueron los siguientes:

Tabla 22
Pre-Test – Indicador de Performance - Frecuencia de tiempos de respuesta al grupo experimental

Grupo Experimental		
Xi	**Frecuencia**	**Porcentaje**
1	0	0.000%
3	8	72.727%
5	3	27.273%
10	0	0.000%
Total	**11**	**100.000%**

Fuente: Pre-test aplicado a los administradores de Red de Petroperú-Iquitos

Fecha: 2014-12-14

Luego, tras contrastar los resultados previos con el Cuadro de categorización de performance para dimensión metodológica de Evaluación, se obtuvieron las siguientes categorías de tiempo de respuesta:

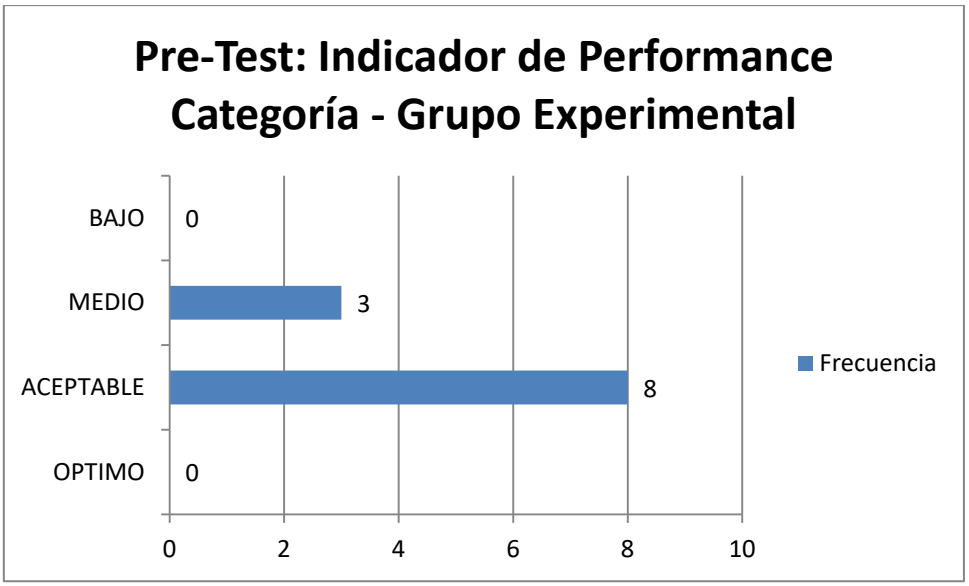
Tabla 23
Pre-Test – Indicador de Performance - Categoría de tiempos de respuesta al grupo experimental

Categoría - Grupo Experimental		
Categoría	**Frecuencia**	**Porcentaje**
OPTIMO	0	0.000%
ACEPTABLE	8	72.727%
MEDIO	3	27.273%
BAJO	0	0.000%
Total	**11**	**100.000%**

Fuente: Tabla 22

Fecha: 2014-12-14

Gráfico 12: Pre-Test - Categoría de tiempos de respuesta al grupo experimental



Fuente: Tabla 23

Fecha: 2014-12-14

Los resultados obtenidos por categorías para medir el tiempo de respuesta en minutos (Xi) que toma atender requerimientos según el proceso actual de monitoreo de las cuentas de usuarios y sus privilegios fueron los siguientes:

En la categoría **BAJO** no se encontraron sistemas y servicios que requieran la atención de sus requerimientos con tiempos de respuesta entre 10 a más minutos. Esto representa el 0% de los resultados.

En la categoría **MEDIO** se encontraron 3 sistemas y servicios que requieran la atención de sus requerimientos con tiempos de respuesta entre 4 a 9 minutos. Esto representa el 27.273% de los resultados.

En la categoría **ACEPTABLE** se encontraron 8 sistemas y servicios que requieran la atención de sus requerimientos con tiempos de respuesta entre 2 a 3 minutos. Esto representa el 72.727% de los resultados.

En la categoría **OPTIMO** no se encontraron sistemas y servicios que requieran la atención de sus requerimientos con tiempos de respuesta entre 0 a 1 minutos. Esto representa el 0% de los resultados. Este indicador demuestra que el proceso de revisión puede ser mejorado.

Tabla 24
Pre-Test – Indicador de Performance - Resultados del Pre-Test al grupo experimental

Resultado Pre-Test – Indicador de Performance - Grupo Experimental			**Estadígrafos**
Categoría	**Frecuencia**	**Porcentaje**	
OPTIMO	0	0.000%	Media: 3.5454 Desv: 0.8727 Cv: 24.6153%
ACEPTABLE	8	72.727%	
MEDIO	3	27.273%	
BAJO	0	0.000%	
Total	**11**	**100.000%**	

Fuente: Tabla 23

Fecha: 2014-12-14

El valor de la media aritmética obtenido por los sistemas y servicios del grupo experimental en el pre-test para medir el tiempo de respuesta en minutos (Xi) que toma atender requerimientos según el proceso actual de monitoreo de las cuentas de usuarios y sus privilegios es de **3.5454**, lo cual nos indica que se encuentra dentro de la categoría **ACEPTABLE** dentro de la escala de la variable dependiente evaluada.

El valor de la desviación estándar es de **0.8727** nos indica que la mayoría de valores se distribuyen a esa distancia a la derecha e izquierda alrededor del promedio.

Finalmente, se observa un coeficiente de variación de **24.6153%** lo cual nos indica un resultado homogéneo.

Relativo a la Seguridad

A los sistemas y servicios que conforman el grupo de estudio se les aplicó el Pre-test, con el propósito de determinar cuan seguro es atender requerimientos según el proceso actual de monitoreo de las cuentas de usuarios y sus privilegios y los resultados fueron los siguientes.

Tabla 25
Pre-Test – Indicador Seguridad - Resultados del Pre-Test al grupo experimental

Servidor	Sistema o Servicio	Autenticación	Control de Conexiones	Encriptación	Categoría
Proxy Server	Linux	SI	SI	2048bits	Aceptable
Mail Server	Linux	SI	SI	2048bits	Aceptable
SMB Server	Linux	SI	NO	2048bits	Medio
Central Server	Linux	SI	NO	2048bits	Medio
Central Server	MySQL	SI	NO	2048bits	Medio
Backup Server	Linux	SI	NO	2048bits	Medio
Syslog Server	Linux	SI	NO	2048bits	Medio
Monitor Server 1	Linux	SI	NO	2048bits	Medio
Monitor Server 1	MySQL	SI	NO	2048bits	Medio
Monitor Server 2	MySQL	SI	NO	2048bits	Medio
Directory Server	Linux	SI	NO	2048bits	Medio

Fuente: Pre-Test Petroperú-Iquitos

Fecha: 2014-12-14

Luego, tras contrastar los resultados previos con el Cuadro de categorización de seguridad para dimensión metodológica de Evaluación, se obtuvieron las siguientes categorías:

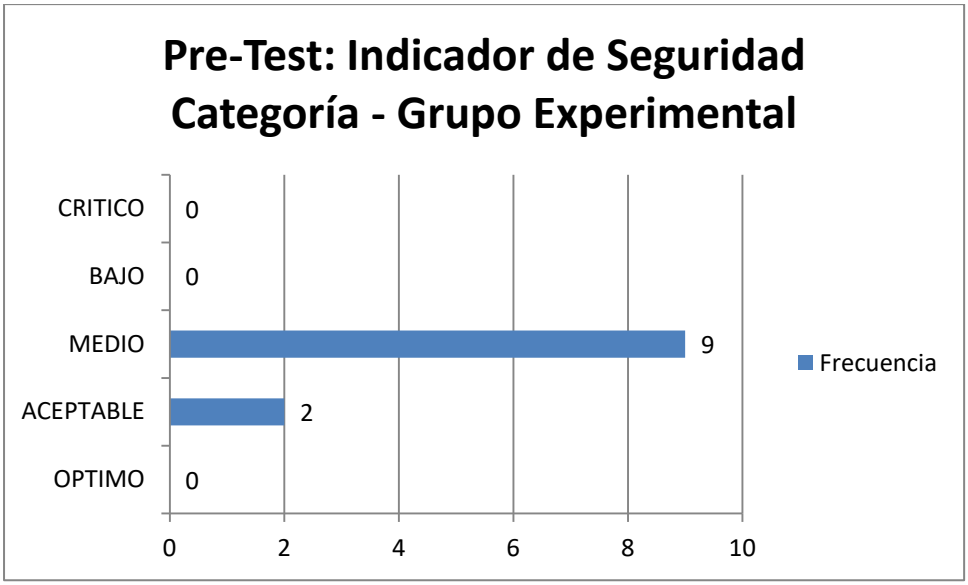
Tabla 26
Pre-Test – Indicador Seguridad - Categoría al grupo experimental

Categoría - Grupo Experimental		
Categoría	**Frecuencia**	**Porcentaje**
OPTIMO	0	0.000%
ACEPTABLE	2	18.182%
MEDIO	9	81.818%
BAJO	0	0.000%
CRITICO	**0**	0.000%
Total	**11**	**100.000%**

Fuente: Tabla 25

Fecha: 2014-12-14

Gráfico 13: Pre-Test – Indicador Seguridad - Categoría al grupo experimental



Fuente: Tabla 26

Fecha: 2014-12-14

Los resultados mostrados en el gráfico, evidencian de manera clara que 9 sistemas y servicios tienen la categoría MEDIO, y 2 sistemas y servicios tienen la categoría ACEPTABLE. Para resumir los resultados obtenidos se elaboró el siguiente cuadro:

Tabla 27
Pre-Test – Indicador de Seguridad - Resultados del Pre-Test al grupo experimental

Resultado Pre-Test – Indicador Seguridad – Grupo Experimental		
Categoría	**Frecuencia**	**Porcentaje**
OPTIMO	0	0.000%
ACEPTABLE	2	18.182%
MEDIO	9	81.818%
BAJO	0	0.000%
CRITICO	0	0.000%
Total	**11**	**100.000%**

Fuente: Tabla 25

Fecha: 2014-12-14

En la presente tabla podemos apreciar que, los resultados obtenidos por categorías para medir el nivel de seguridad en atender requerimientos según el proceso actual de monitoreo de las cuentas de usuarios y sus privilegios fueron los siguientes:

En la categoría **CRÍTICO** no se encontraron sistemas y servicios que no usen mecanismo de autenticación, controles de conexión y que no cifren sus comunicaciones. Esto representa el 0% de los resultados.

En la categoría **BAJO** no se encontraron sistemas y servicios que si usen mecanismos de autenticación, pero que no cuenten con controles de conexión pero si con cifrado menor o igual a 512 bits en sus comunicaciones. Esto representa el 0% de los resultados.

En la categoría **MEDIO** se encontraron 9 sistemas y servicios que si utilizan mecanismos de autenticación, no cuentan con un control de conexiones y utilizan una encriptación mayor o igual a 1024bits y menor o igual a 2048bits. Esto representa el 81.818% de los resultados.

En la categoría **ACEPTABLE** se encontraron 2 sistemas y servicios que si utilizan mecanismos de autenticación, si cuentan con un control de conexiones y utilizan una encriptación mayor o igual a 1024bits y menor o igual a 2048bits. Esto representa el 18.182% de los resultados.

En la categoría **ÓPTIMO** no se encontraron sistemas y servicios que utilicen un mecanismo de autenticación, controles de conexión, y utilicen una encriptación mayor o igual a 4096 bits. Esto representa el 0% de los resultados. Este indicador demuestra que el proceso de revisión puede ser mejorado.

4.1.3. Objetivo 3

Diseñar el protocolo SUMP para mejorar la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux.

Para cumplir este objetivo se realizaron las siguientes actividades

Tabla 28
Actividades para cumplir el objetivo 3

Actividad	Descripción
Elaborar Propuesta de Protocolo	Elaborar una propuesta teórica y tecnológica para el desarrollo de un protocolo para cumplir con el objetivo 3 sobre los sistemas y servicios con prioridad Alta indicados en la Tabla 21

Fuente: Elaboración del investigador

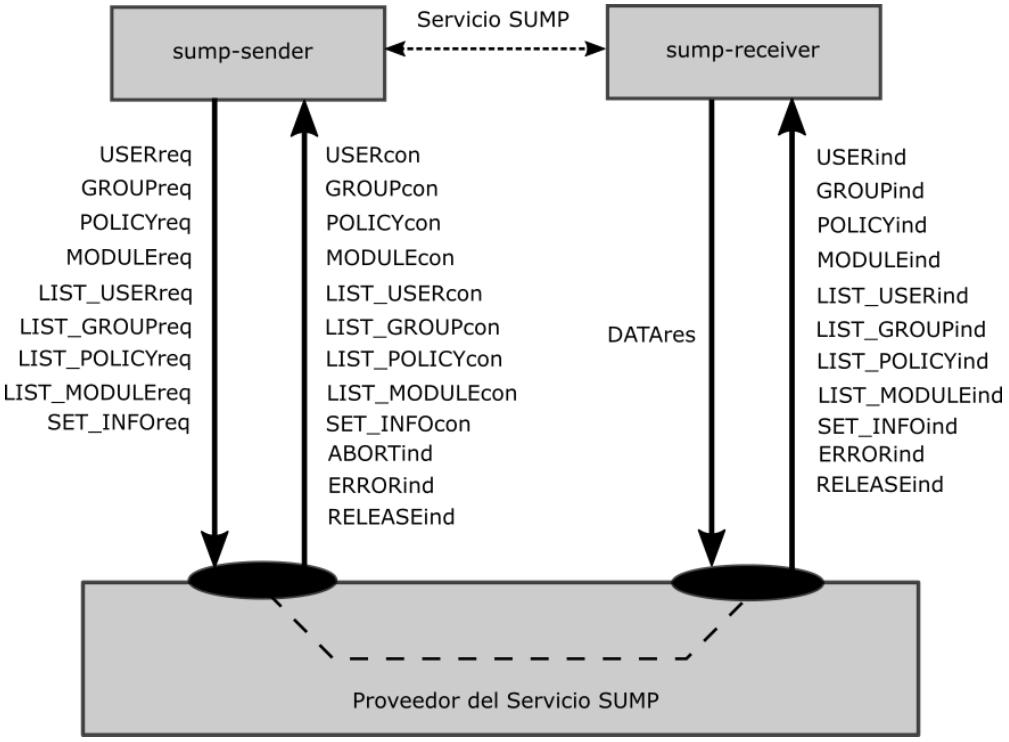
4.1.3.1. Elaborar Propuesta de Protocolo

Para el desarrollo de esta actividad se siguió la metodología de diseño formal propuesta por König que consiste en trabajar con 3 especificaciones (servicio, protocolo e implementación). Así mismo, dentro de la especificación de la implementación se ha considerado la metodología heurística también propuesta por König para brindar un entregable más preciso a nivel técnico mediante el uso del formato RFC.

4.1.3.1.1. Propuesta de Especificación de Servicio

Esta propuesta describe la interacción del servicio SUMP entre los usuarios del servicio, los proveedores y sus interfaces. Las técnicas de descripción formal que se utilizaron fueron los diagramas TS (time-sequence) para representar las secuencias lógicas del protocolo. Así mismo, se consideró el uso de diagramas de estado para explicar sus diferentes fases. Finalmente se ha utilizado el lenguaje de modelado Level S para describir con más detalle los elementos de esta especificación.

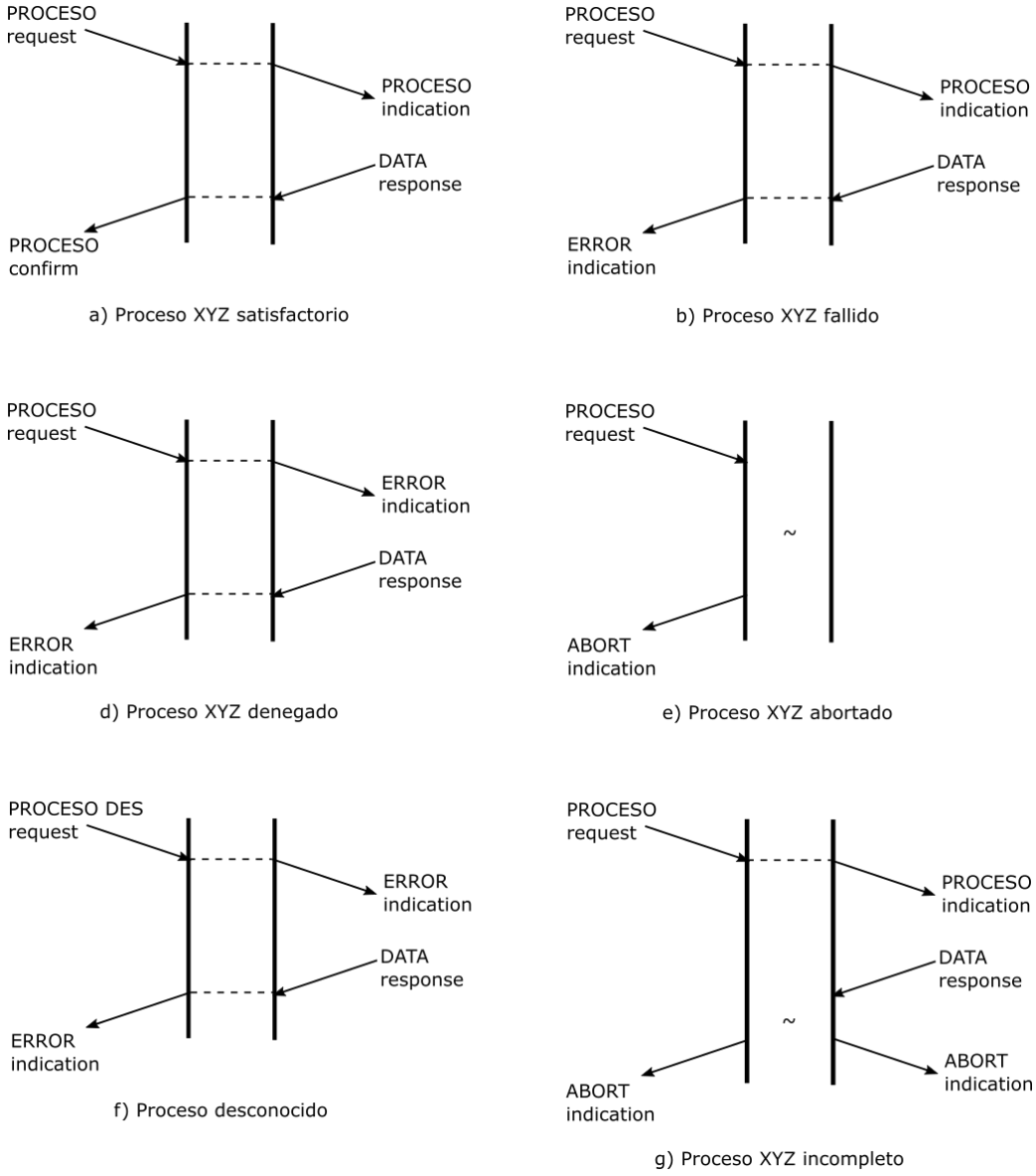
Gráfico 14: Diagrama de Especificación del servicio SUMP



Fuente: Elaboración del investigador

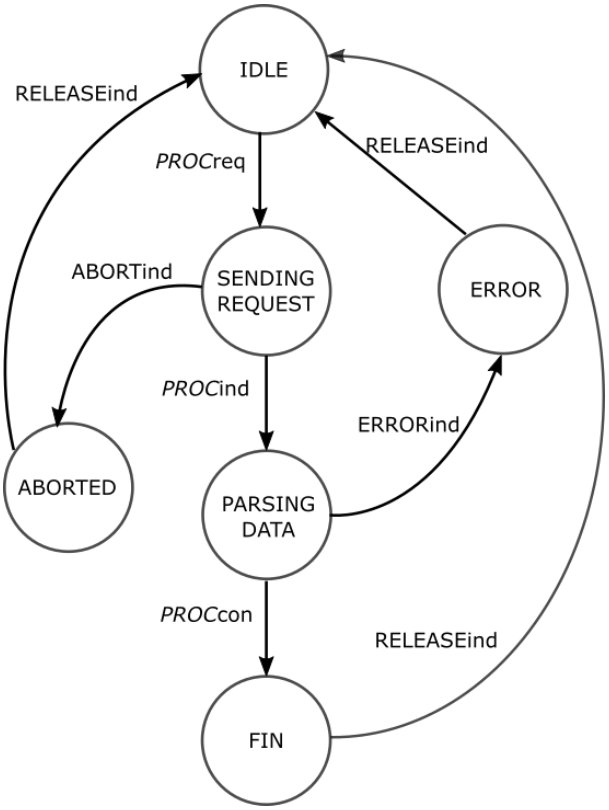
A continuación se detalla la propuesta operativa del protocolo

Gráfico 15: Diagrama TS (time sequence) del protocolo SUMP



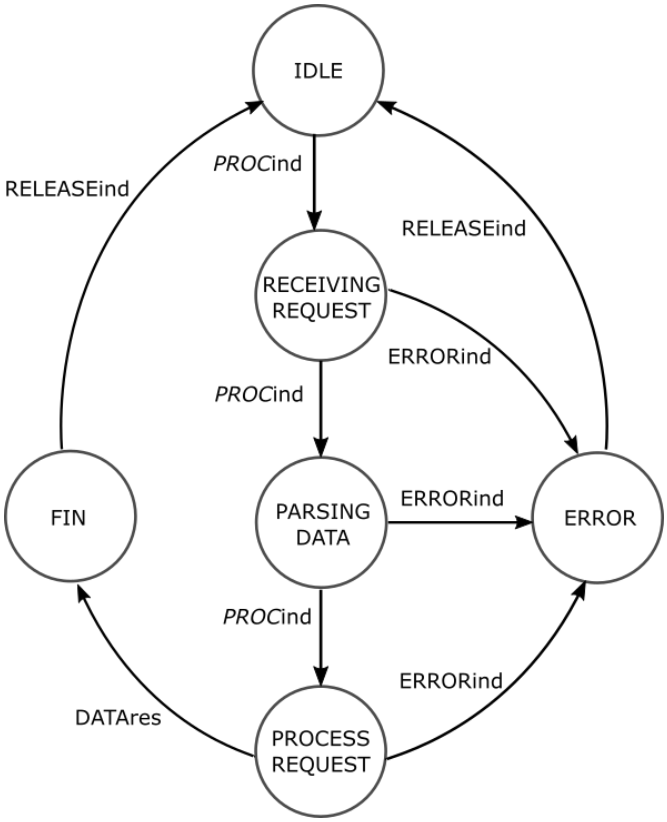
Fuente: Elaboración del investigador

Gráfico 16: Diagrama de estado sump-sender



Fuente: Elaboración del investigador

Gráfico 17: Diagrama de estado sump-receiver



Fuente: Elaboración del investigador

specification SUMP {
service SUMP-MANAGER:
requested USERreq(key: **string**, mod: **string**, user: **string**),
GROUPreq(key: **string**, mod: **string**, group: **string**),
POLICYreq(key: **string**, mod: **string**, policy: **string**),
MODULEreq(key: **string**, mod: **string**),
LIST_USERreq(key: **string**, mod: **string**),
LIST_GROUPreq(key: **string**, mod: **string**),
LIST_POLICYreq(key: **string**, mod: **string**),
LIST_MODULEreq(key: **string**)
SET_INFOreq(key: **string**, mod:**string**, info:**string**)
responded USERcon(data: **string**),
GROUPcon(data: **string**),
POLICYcon(data: **string**),
MODULEcon(data: **string**),
LIST_USERcon(data: **string**),
LIST_GROUPcon(data: **string**),
LIST_POLICYcon(data: **string**),
LIST_MODULEcon(data: **string**),
SET_INFOcon(data: **string**)
ABORTind(code: **integer**, msg: **string**),

```
        ERRORind(code: integer, msg: string),
        RELEASEind(req: string, res: string),
        DATAres(data: array [ ] of byte)

sap sump-sender{
signal response_ok, parse_done, abort, error
phase REQUEST USER: USERreq, LIST_USERreq, RELEASEind,
                    ABORTind, ERRORind;
      REQUEST GROUP: GROUPreq, LIST_GROUPreq,
                    RELEASEind, ABORTind, ERRORind;
      REQUEST MODULE: MODULEreq, LIST_MODULEreq,
                    RELEASEind, ABORTind, ERRORind;
      REQUEST POLICY: POLICYreq, LIST_POLICYreq, RELEASEind,
                    ABORTind, ERRORind;
      REQUEST INFO: SET_INFOreq, SET_INFOind,
                    ABORTind, ERRORind;
      PARSER: USERind, GROUPind,
                    MODULEind, POLICYind, LIST_USERind,
                    LIST_GROUPind, LIST_MODULEind,
                    LIST_POLICYind, USERcon, LIST_USERcon,
                    GROUPcon, LIST_GROUPcon, MODULEcon,
                    LIST_MODULEcon, POLICYcon, SET_INFOcon,
                    LIST_POLICYcon, RELEASEind, ERRORind;
wait event {
      USERreq(key, mod, user):
            set REQUEST USER
            ➥sump-receiver.USERind
            set PARSER
            wait event{
              response_ok:
                  wait event{
                    parse_done: respond USERcon(res)
                                respond RELEASEind(req,res) |
                    error n: respond ERRORind(n, “ERROR”)
                          respond RELEASEind(req,res)
                  }|
              error: respond ERRORind(600,”INTERNAL ERROR”)
                     respond RELEASEind(req,res)
            } |
      LIST_USERreq(key, mod):
            set REQUEST USER
            ➥sump-receiver.LIST_USERind
            set PARSER
            wait event{
              response_ok:
                  wait event{
                    parse_done: respond LIST_USERcon(res)
```

```
                                    respond RELEASEind(req,res) |
                        error n: respond ERRORind(n, “ERROR”)
                                    respond RELEASEind(req,res)
                        }|
                error: respond ERRORind(600,”INTERNAL ERROR”)
                        respond RELEASEind(req,res)
        } |
GROUPreq(key, mod, group):
        set REQUEST GROUP
        ➥sump-receiver.GROUPind
        set PARSER
        wait event{
          response_ok:
                wait event{
                  parse_done: respond GROUPcon(res)
                                    respond RELEASEind(req,res) |
                  error n: respond ERRORind(n, “ERROR”)
                                    respond RELEASEind(req,res)
                }|
          error: respond ERRORind(600,”INTERNAL ERROR”)
                  respond RELEASEind(req,res)
        } |
LIST_GROUPreq(key, mod):
        set REQUEST GROUP
        ➥sump-receiver.LIST_GROUPind
        set PARSER
        wait event{
          response_ok:
                wait event{
                  parse_done: respond LIST_GROUPcon(res)
                                    respond RELEASEind(req,res) |
                  error n: respond ERRORind(n, “ERROR”)
                                    respond RELEASEind(req,res)
                }|
          error: respond ERRORind(600,”INTERNAL ERROR”)
                  respond RELEASEind(req,res)
        } |
MODULEreq(key, mod):
        set REQUEST MODULE
        ➥sump-receiver.MODULEind
        set PARSER
        wait event{
          response_ok:
                wait event{
                  parse_done: respond MODULEcon(res)
                                    respond RELEASEind(req,res) |
                  error n: respond ERRORind(n, “ERROR”)
```

```
                    respond RELEASEind(req,res)
                }|
            error: respond ERRORind(600,"INTERNAL ERROR")
                    respond RELEASEind(req,res)
        } |
LIST_MODULEreq(key):
        set REQUEST MODULE
        ➥sump-receiver.LIST_MODULEind
        set PARSER
        wait event{
            response_ok:
                wait event{
                    parse_done: respond LIST_MODULEcon(res)
                                respond RELEASEind(req,res) |
                    error n: respond ERRORind(n, "ERROR")
                            respond RELEASEind(req,res)
                }|
            error: respond ERRORind(600,"INTERNAL ERROR")
                    respond RELEASEind(req,res)
        } |
POLICYreq(key, mod, policy):
        set REQUEST POLICY
        ➥sump-receiver.POLICYind
        set PARSER
        wait event{
            response_ok:
                wait event{
                    parse_done: respond POLICYcon(res)
                                respond RELEASEind(req,res) |
                    error n: respond ERRORind(n, "ERROR")
                            respond RELEASEind(req,res)
                }|
            error: respond ERRORind(600,"INTERNAL ERROR")
                    respond RELEASEind(req,res)
        } |
LIST_POLICYreq(key, mod):
        set REQUEST POLICY
        ➥sump-receiver.LIST_POLICYind
        set PARSER
        wait event{
            response_ok:
                wait event{
                    parse_done: respond LIST_POLICYcon(res)
                                respond RELEASEind(req,res) |
                    error n: respond ERRORind(n, "ERROR")
                            respond RELEASEind(req,res)
                }|
```

```
                    error: respond ERRORind(600,"INTERNAL ERROR")
                           respond RELEASEind(req,res)
                    } |
            SET_INFOreq(key, mod, info):
                    set REQUEST INFO
                    ➥sump-receiver.SET_INFOind
                    set PARSER
                    wait event{
                      response_ok:
                            wait event{
                              parse_done: respond SET_INFOcon(res)
                                          respond RELEASEind(req,res) |
                              error n: respond ERRORind(n, "ERROR")
                                       respond RELEASEind(req,res)
                            }|
                      error: respond ERRORind(600,"INTERNAL ERROR")
                             respond RELEASEind(req,res)
                    } |
            abort: respond ABORTind(602,"REQUEST ABORTED")
                   respond RELEASEind(req,res)
} // wait-event
} // sap sump-sender

sap sump-receiver{
signal request_ok, response_ok, error
phase RESPONSE USER: USERind, LIST_USERind,
                     RELEASEind, ERRORind;
      RESPONSE GROUP: GROUPind, LIST_GROUPind,
                     RELEASEind, ERRORind;
      RESPONSE MODULE: MODULEind, LIST_MODULEind,
                     RELEASEind, ERRORind;
      RESPONSE POLICY: POLICYind, LIST_POLICYind,
                     RELEASEind, ERRORind;
      RESPONSE INFO: SET_INFOind,
                     RELEASEind, ERRORind;
      PARSER: USERind, GROUPind,
                   MODULEind, POLICYind, LIST_USERind,
                   LIST_GROUPind, LIST_MODULEind,
                   LIST_POLICYind, DATAres, RELEASEind,
                   ERRORind
wait event {
      ➥sump-sender.USERreq:
            set PARSER
            wait event{
              request_ok:
                    set RESPONSE USER
                    respond USERind(req)
```

```
                wait event{
                  response_ok: respond DATAres(res)
                               respond RELEASEind(req,res) |
                  error n: respond ERRORind(req, n)
                           respond DATAres(res)
                           respond RELEASEind(req,res)
                } |
          error m: respond ERRORind(req, m)
                   respond DATAres(res)
                   respond RELEASEind(req,res)
        } |
↪sump-sender.LIST_USERreq:
        set PARSER
        wait event{
          request_ok:
                set RESPONSE USER
                respond LIST_USERind(req)
                wait event{
                  response_ok: respond DATAres(res)
                               respond RELEASEind(req,res) |
                  error n: respond ERRORind(req, n)
                           respond DATAres(res)
                           respond RELEASEind(req,res)
                } |
          error m: respond ERRORind(req, m)
                   respond DATAres(res)
                   respond RELEASEind(req,res)
        } |
↪sump-sender.GROUPreq:
        set PARSER
        wait event{
          request_ok:
                set RESPONSE GROUP
                respond GROUPind(req)
                wait event{
                  response_ok: respond DATAres(res)
                               respond RELEASEind(req,res) |
                  error n: respond ERRORind(req, n)
                           respond DATAres(res)
                           respond RELEASEind(req,res)
                } |
          error m: respond ERRORind(req, m)
                   respond DATAres(res)
                   respond RELEASEind(req,res)
        } |
↪sump-sender.LIST_GROUPreq:
        set PARSER
```

```
        wait event{
          request_ok:
                set RESPONSE GROUP
                respond LIST_GROUPind(req)
                wait event{
                  response_ok: respond DATAres(res)
                               respond RELEASEind(req,res) |
                  error n: respond ERRORind(req, n)
                           respond DATAres(res)
                           respond RELEASEind(req,res)
                } |
          error m: respond ERRORind(req, m)
                   respond DATAres(res)
                   respond RELEASEind(req,res)
        } |
➥sump-sender.MODULEreq:
        set PARSER
        wait event{
          request_ok:
                set RESPONSE MODULE
                respond MODULEind(req)
                wait event{
                  response_ok: respond DATAres(res)
                               respond RELEASEind(req,res) |
                  error n: respond ERRORind(req, n)
                           respond DATAres(res)
                           respond RELEASEind(req,res)
                } |
          error m: respond ERRORind(req, m)
                   respond DATAres(res)
                   respond RELEASEind(req,res)
        } |
➥sump-sender.LIST_MODULEreq:
        set PARSER
        wait event{
          request_ok:
                set RESPONSE MODULE
                respond LIST_MODULEind(req)
                wait event{
                  response_ok: respond DATAres(res)
                               respond RELEASEind(req,res) |
                  error n: respond ERRORind(req, n)
                           respond DATAres(res)
                           respond RELEASEind(req,res)
                } |
          error m: respond ERRORind(req, m)
                   respond DATAres(res)
```

```
                    respond RELEASEind(req,res)
        } |
➥sump-sender.POLICYreq:
        set PARSER
        wait event{
          request_ok:
                set RESPONSE POLICY
                respond POLICYind(req)
                wait event{
                  response_ok: respond DATAres(res)
                               respond RELEASEind(req,res) |
                  error n: respond ERRORind(req, n)
                           respond DATAres(res)
                           respond RELEASEind(req,res)
                } |
          error m: respond ERRORind(req, m)
                   respond DATAres(res)
                   respond RELEASEind(req,res)
        } |
➥sump-sender.LIST_POLICYreq:
        set PARSER
        wait event{
          request_ok:
                set RESPONSE POLICY
                respond LIST_POLICYind(req)
                wait event{
                  response_ok: respond DATAres(res)
                               respond RELEASEind(req,res) |
                  error n: respond ERRORind(req, n)
                           respond DATAres(res)
                           respond RELEASEind(req,res)
                } |
          error m: respond ERRORind(req, m)
                   respond DATAres(res)
                   respond RELEASEind(req,res)
        } |
➥sump-sender.SET_INFOreq:
        set PARSER
        wait event{
          request_ok:
                set RESPONSE INFO
                respond SET_INFOind(req)
                wait event{
                  response_ok: respond DATAres(res)
                               respond RELEASEind(req,res) |
                  error n: respond ERRORind(req, n)
                           respond DATAres(res)
```

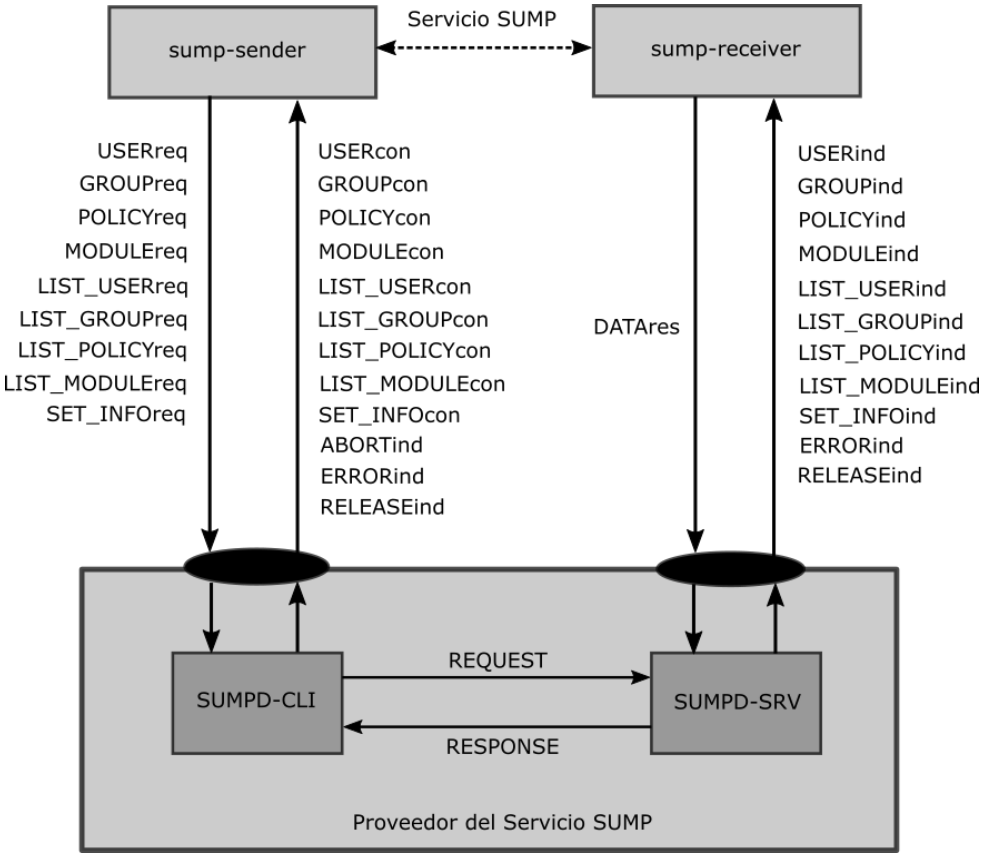
```
                                    respond RELEASEind(req,res)
                            } |
                    error m: respond ERRORind(req, m)
                             respond DATAres(res)
                             respond RELEASEind(req,res)
                    } |
            error e:  respond ERRORind(req, e)
                      respond DATAres(res)
                      respond RELEASEind(req,res)
    } // wait-event
    } // sap sump-receiver

} // SUMP
```

4.1.3.1.2. Propuesta de Especificación de Protocolo

Esta propuesta describe la interacción de las entidades pares SUMP-CLI y SUMP-SRV, el orden de sus funciones y el formato de sus PDUs. Se ha utilizado el lenguaje de modelado Level P como técnica de descripción formal del protocolo SUMP.

Gráfico 18: Diagrama de Especificación del Protocolo SUMP



Fuente: Elaboración del investigador

```
specification SUMP {
service SUMP-MANAGER:
message {
   REQ: struct(                                    // REQUEST PDU
                method: char(33),
                protocol: char(12),
                version char(6),
                header: struct varBindList (
                              info: sequence of struct varBind(
                                        name: char(256),
                                        value: [ ] byte
                              ),
                              nelements: integer
                       )
                )
   RES: struct(                                    // RESPONSE PDU
                method: char(33),
                protocol: char(12),
                version char(6),
                status: struct sc(code: char(4), msg: [ ] byte),
                header: struct varBindList (
                              info: sequence of struct varBind(
                                        name: char(256),
                                        value: [ ] byte
                              ),
                              nelements: integer
                       ),
                data: [ ] byte
                )
protocol
   mgmt_users: SUMPD-CLI.request_users ⇔ SUMPD-SRV.response_users
   mgmt_group: SUMPD-CLI.request_group ⇔ SUMPD-SRV.response_group
   mgmt_modu: SUMPD-CLI.request_modu ⇔ SUMPD-SRV.response_modu
   mgmt_policy: SUMPD-CLI.request_policy ⇔ SUMPD-SRV.response_policy
   mgmt_info: SUMPD-CLI.request_info ⇔ SUMPD-SRV.response_info

entity SUMPD-CLI
   sap sump-sender
   signal IDLE, SENDING REQUEST, PARSE, ERROR, ABORTED,FIN
   wait event {
          USERreq or LIST_USERreq: request_users |
          GROUPreq or LIST_GROUPreq: request_group |
          MODULEreq or LIST_MODULEreq: request_modu |
```

```
            POLICYreq or LIST_POLICYreq: request_policy |
            SET_INFOreq: request_info
    }

entity SUMPD-SRV
    sap sump-receiver
    signal IDLE, RECEIVING REQUEST, PARSE, PROCESS REQUEST,
            ERROR, FIN
    wait event {
            REQ ← sump-sender
                    and (REQ.method = “GET_USER”
                    or REQ.method = “GET_LIST_USER”): responde_users |
            REQ ← sump-sender
                    and (REQ.method = “GET_GROUP”
                    or REQ.method = “GET_LIST_GROUP”): responde_group |
            REQ ← sump-sender
                    and (REQ.method = “GET_MODULE”
                    or REQ.method = “GET_LIST_MODULE”): responde_modu |
            REQ ← sump-sender
                    and (REQ.method = “GET_POLICY”
                    or REQ.method = “GET_LIST_POLICY”): responde_policy |
            REQ ← sump-sender
                    and (REQ.method = “SET_INFO”): responde_info

    }

protocol part request_users (peer sump-sender)      // Solicitud de usuario
var err_code: integer init(0)
var err_msg: [ ] byte
timer t: 0..? ms                                    // Cronometro
set SENDING REQUEST                                 // Fase de envío
wait event {
USERreq or LIST_USERreq:                            // Trigger event
    begin
            coding_request(REQ)                     // Codificación
            REQ →sump-receiver                      // Enviando datos
            start t                                 // Iniciar cronometro
            set PARSER                              // Fase Parser
            wait event{
              request_ok:
                    wait event{
                    RES ← sump-receiver and
                            RES.header.nelements > 0:
                            reset t
                            if (RES.method = “GET_USER”){
                                    respond USERcon(RES)
                            }
```

```
                    else respond LIST_USERcon(RES)
                    set FIN
                    respond RELEASEind(REQ,RES)
                    set IDLE |
                RES ← sump-receiver and
                    RES.status.code >= 400:
                    reset t
                    err_code = RES.status.code
                    err_msg = RES.status.msg
                    respond ERRORind(err_code, err_msg)
                    respond RELEASEind(REQ,RES) |
                timeout t: set ERROR
                    reset t
                    err_code = 601
                    err_msg = “TIMEOUT ON ” + REQ.method
                    respond ERRORind(err_code, err_msg)
                    respond RELEASEind(REQ,RES)
                    set IDLE
                } | // wait-event request_ok
            error: set ERROR
                reset t
                err_code = 600
                err_msg = “INTERNAL ERROR ON ” + REQ.method
                respond ERRORind(err_code, err_msg)
                respond RELEASEind(REQ,RES)
                set IDLE
            }
    end|
abort:
  begin
    set ABORTED
    err_code = 602
    err_msg = “REQUEST ABORTED ON ” + REQ.method
    respond ABORTind(err_code, err_msg)
    respond RELEASEind(REQ,RES)
    set IDLE
  end
} // wait-event main

protocol part request_group (peer sump-sender)
var err_code: integer init(0)
var err_msg: [ ] byte
timer t: 0..? ms                                    // Cronometro
set SENDING REQUEST                                 // Fase de envío
wait event {
GROUPreq or LIST_GROUPreq:                          // Trigger event
    begin
```

```
        coding_request(REQ)                    // Codificación
        REQ →sump-receiver                     // Enviando datos
        start t                                // Iniciar cronometro
        set PARSER                             // Fase PARSER
        wait event{
          request_ok:
                wait event{
                RES ← sump-receiver and
                        RES.header.nelements > 0:
                        reset t
                        if (RES.method = "GET_GROUP"){
                                respond GROUPcon(RES)
                        }
                        else respond LIST_GROUPcon(RES)
                        set FIN
                        respond RELEASEind(REQ,RES)
                        set IDLE |
                RES ← sump-receiver and
                        RES.status.code >= 400:
                        reset t
                        err_code = RES.status.code
                        err_msg = RES.status.msg
                        respond ERRORind(err_code, err_msg)
                        respond RELEASEind(REQ,RES) |
                timeout t: set ERROR
                        reset t
                        err_code = 601
                        err_msg = "TIMEOUT ON " + REQ.method
                        respond ERRORind(err_code, err_msg)
                        respond RELEASEind(REQ,RES)
                        set IDLE
                } | // wait-event request_ok
        error: set ERROR
                reset t
                err_code = 600
                err_msg = "INTERNAL ERROR ON " + REQ.method
                respond ERRORind(err_code, err_msg)
                respond RELEASEind(REQ,RES)
                set IDLE
        }
    end|
abort:
  begin
    set ABORTED
    err_code = 602
    err_msg = "REQUEST ABORTED ON " + REQ.method
    respond ABORTind(err_code, err_msg)
```

```
    respond RELEASEind(REQ,RES)
    set IDLE
   end
} // wait-event main


protocol part request_modu (peer sump-sender)
var err_code: integer init(0)
var err_msg: [ ] byte
timer t: 0..? ms                                    // Cronometro
set SENDING REQUEST                                 // Fase de envío
wait event {
MODULEreq or LIST_MODULEreq:                        // Trigger event
    begin
            coding_request(REQ)                     // Codificación
            REQ →sump-receiver                      // Enviando datos
            start t                                 // Iniciar cronometro
            set PARSER                              // Fase Parser
            wait event{
              request_ok:
                    wait event{
                    RES ← sump-receiver and
                            RES.header.nelements > 0:
                            reset t
                            if (RES.method = "GET_MODULE"){
                                    respond MODULEcon(RES)
                            }
                            else respond LIST_MODULEcon(RES)
                            set FIN
                            respond RELEASEind(REQ,RES)
                            set IDLE |
                    RES ← sump-receiver and
                            RES.status.code >= 400:
                            reset t
                            err_code = RES.status.code
                            err_msg = RES.status.msg
                            respond ERRORind(err_code, err_msg)
                            respond RELEASEind(REQ,RES) |
                    timeout t: set ERROR
                            reset t
                            err_code = 601
                            err_msg = "TIMEOUT ON " + REQ.method
                            respond ERRORind(err_code, err_msg)
                            respond RELEASEind(REQ,RES)
                            set IDLE
                    } | // wait-event request_ok
            error: set ERROR
```

```
                    reset t
                    err_code = 600
                    err_msg = “INTERNAL ERROR ON ” + REQ.method
                    respond ERRORind(err_code, err_msg)
                    respond RELEASEind(REQ,RES)
                    set IDLE
            }
    end|
abort:
  begin
    set ABORTED
    err_code = 602
    err_msg = “REQUEST ABORTED ON ” + REQ.method
    respond ABORTind(err_code, err_msg)
    respond RELEASEind(REQ,RES)
    set IDLE
  end
} // wait-event main


protocol part request_policy (peer sump-sender)
var err_code: integer init(0)
var err_msg: [ ] byte
timer t: 0..? ms                                    // Cronometro
set SENDING REQUEST                                 // Fase de envío
wait event {
POLICYreq or LIST_POLICYreq:                        // Trigger event
    begin
            coding_request(REQ)                     // Codificación
            REQ →sump-receiver                      // Enviando datos
            start t                                 // Iniciar cronometro
            set PARSER                              // Fase Parser
            wait event{
              request_ok:
                    wait event{
                    RES ← sump-receiver and
                            RES.header.nelements > 0:
                            reset t
                            if (RES.method = “GET_POLICY”){
                                    respond USERcon(RES)
                            }
                            else respond LIST_POLICYcon(RES)
                            set FIN
                            respond RELEASEind(REQ,RES)
                            set IDLE |
                    RES ← sump-receiver and
                            RES.status.code >= 400:
```

```
                    reset t
                    err_code = RES.status.code
                    err_msg = RES.status.msg
                    respond ERRORind(err_code, err_msg)
                    respond RELEASEind(REQ,RES) |
                timeout t: set ERROR
                    reset t
                    err_code = 601
                    err_msg = “TIMEOUT ON ” + REQ.method
                    respond ERRORind(err_code, err_msg)
                    respond RELEASEind(REQ,RES)
                    set IDLE
                } | // wait-event request_ok
            error: set ERROR
                reset t
                err_code = 600
                err_msg = “INTERNAL ERROR ON ” + REQ.method
                respond ERRORind(err_code, err_msg)
                respond RELEASEind(REQ,RES)
                set IDLE
            }
    end|
abort:
  begin
    set ABORTED
    err_code = 602
    err_msg = “REQUEST ABORTED ON ” + REQ.method
    respond ABORTind(err_code, err_msg)
    respond RELEASEind(REQ,RES)
    set IDLE
  end
} // wait-event main

protocol part request_info (peer sump-sender)
var err_code: integer init(0)
var err_msg: [ ] byte
timer t: 0..? ms                                    // Cronometro
set SENDING REQUEST                                 // Fase de envío
wait event {
SET_INFOreq:                                        // Trigger event
    begin
        coding_request(REQ)                         // Codificación
        REQ →sump-receiver                          // Enviando datos
        start t                                     // Iniciar cronometro
        set PARSER                                  // Fase Parser
        wait event{
          request_ok:
```

```
                    wait event{
                    RES ← sump-receiver and
                            RES.header.nelements > 0:
                            reset t
                            if (RES.method = “SET_INFO”){
                                    respond SET_INFOcon(RES)
                            }
                            set FIN
                            respond RELEASEind(REQ,RES)
                            set IDLE |
                    RES ← sump-receiver and
                            RES.status.code >= 400:
                            reset t
                            err_code = RES.status.code
                            err_msg = RES.status.msg
                            respond ERRORind(err_code, err_msg)
                            respond RELEASEind(REQ,RES) |
                    timeout t: set ERROR
                            reset t
                            err_code = 601
                            err_msg = “TIMEOUT ON ” + REQ.method
                            respond ERRORind(err_code, err_msg)
                            respond RELEASEind(REQ,RES)
                            set IDLE
                    } | // wait-event request_ok
            error: set ERROR
                    reset t
                    err_code = 600
                    err_msg = “INTERNAL ERROR ON ” + REQ.method
                    respond ERRORind(err_code, err_msg)
                    respond RELEASEind(REQ,RES)
                    set IDLE
            }
    end|
abort:
  begin
    set ABORTED
    err_code = 602
    err_msg = “REQUEST ABORTED ON ” + REQ.method
    respond ABORTind(err_code, err_msg)
    respond RELEASEind(REQ,RES)
    set IDLE
  end
} // wait-event main
```

```
protocol part response_users (peer sump-receiver)
var err_code: integer init(0)
var err, err_msg: [ ] byte
var err_matrix: array[15] of record(sign: [ ] byte, code: integer, msg: [ ] byte)
err_matrix = {
   {“req_too_long”,413,”REQUEST TOO LONG”},
   {“too_many_reqs”,429,”TOO MANY REQUEST”},
   {“heaer_too_large”,431,”HEADER TOO LARGE”},
   {“bad_header”,432,”BAD HEADER”},
   {“bad_request”,400,”BAD REQUEST”},
   {“method_not_allow”,405,”METHOD NOT ALLOW”},
   {“format_unsupported”,406,”FORMAT NOT SUPPORTED”},
   {“not_implemented”,501,”NOT IMPLEMENTED”},
   {“ver_not_supported”,505,”VERSION NOT SUPPORTED”},
   {“unauthorized”,401,”UNAUTHORIZED”},
   {“forbidden”,403,”FORBIDDEN”},{“not_found”,404,”NOT FOUND”},
   {“internal”,500,”INTERNAL ERROR”},{“unavailable”,503,”UNAVAILABLE”},
   {“timeout”,504,”TIMEOUT”},{“disk_full”,507,”NOT ENOUGH SPACE”}
}
timer t: 0..? ms                                        // Cronometro
set RECEIVING REQUEST
wait event {
REQ ← sump-sender and                                   // Trigger Event
   (REQ.method = “GET_USER” or REQ.method = “GET_LIST_USER”):
   begin
         set PARSER
         wait event{
           request_ok:
               set PROCESS REQUEST
               start t
               if (REQ.method = “GET_USER”)
                     respond USERind(REQ)
               if (REQ.method = “GET_LIST_USER”)
                     respond LIST_USERind(REQ)
               wait event{
                 RES ← sump-receiver:
                     coding_response(RES)
                     RES→sump-sender
                     reset t
                     set FIN
                     respond RELEASEind(REQ,RES)
                     set IDLE |
                 error err:     // Processing Error
                     reset t
                     err_code = get_proc_err_code_from_matrix(err)
                     err_msg = get_ proc_err_msg_from_matrix(err_code)
                     respond ERRORind(REQ, err_code)
```

```
                    set ERROR
                    var ERR: RES
                    ERR.status.code = err_code
                    ERR.status.msg = err_msg
                    coding_response(ERR)
                    ERR→sump-sender
                    respond RELEASEind(REQ,ERR)
                    set IDLE
                }|
            error perr:   // Parsing Error
                err_code = get_parse_err_code_from_matrix(perr)
                err_msg = get_ parse_err_msg_from_matrix(err_code)
                respond ERRORind(REQ, err_code)
                set ERROR
                var ERR: RES
                ERR.status.code = err_code
                ERR.status.msg = err_msg
                coding_response(ERR)
                ERR→sump-sender
                respond RELEASEind(REQ,ERR)
                set IDLE
        }
    end|
(req_too_long or too_many_reqs or header_too_large or bad_header) err:
    begin
        err_code = get_recv_err_code_from_matrix(err)
        err_msg = get_ recv_err_msg_from_matrix(err_code)
        respond ERRORind(REQ, err_code)
        set ERROR
        var ERR: RES
        ERR.status.code = err_code
        ERR.status.msg = err_msg
        coding_response(ERR)
        ERR→sump-sender
        respond RELEASEind(REQ,ERR)
        set IDLE
    end
} // wait-event main

protocol part response_group (peer sump- receiver)
var err_code: integer init(0)
var err, err_msg: [ ] byte
var err_matrix: array[15] of record(sign: [ ] byte, code: integer, msg: [ ] byte)
err_matrix = {
    {“req_too_long”,413,”REQUEST TOO LONG”},
    {“too_many_reqs”,429,”TOO MANY REQUEST”},
    {“heaer_too_large”,431,”HEADER TOO LARGE”},
```

```
    {“bad_header”,432,”BAD HEADER”},
    {“bad_request”,400,”BAD REQUEST”},
    {“method_not_allow”,405,”METHOD NOT ALLOW”},
    {“format_unsupported”,406,”FORMAT NOT SUPPORTED”},
    {“not_implemented”,501,”NOT IMPLEMENTED”},
    {“ver_not_supported”,505,”VERSION NOT SUPPORTED”},
    {“unauthorized”,401,”UNAUTHORIZED”},
    {“forbidden”,403,”FORBIDDEN”},{“not_found”,404,”NOT FOUND”},
    {“internal”,500,”INTERNAL ERROR”},{“unavailable”,503,”UNAVAILABLE”},
    {“timeout”,504,”TIMEOUT”},{“disk_full”,507,”NOT ENOUGH SPACE”}
}
timer t: 0..? ms                                        // Cronometro
set RECEIVING REQUEST
wait event {
REQ ← sump-sender and                                   // Trigger Event
   (REQ.method = “GET_GROUP” or REQ.method = “GET_LIST_GROUP”):
   begin
          set PARSER
          wait event{
            request_ok:
                set PROCESS REQUEST
                start t
                if (REQ.method = “GET_GROUP”)
                        respond GROUPind(REQ)
                if (REQ.method = “GET_LIST_GROUP”)
                        respond LIST_GROUPind(REQ)
                wait event{
                  RES ← sump-receiver:
                        coding_response(RES)
                        RES→sump-sender
                        reset t
                        set FIN
                        respond RELEASEind(REQ,RES)
                        set IDLE |
                  error err:     // Processing Error
                        reset t
                        err_code = get_proc_err_code_from_matrix(err)
                        err_msg = get_ proc_err_msg_from_matrix(err_code)
                        respond ERRORind(REQ, err_code)
                        set ERROR
                        var ERR: RES
                        ERR.status.code = err_code
                        ERR.status.msg = err_msg
                        coding_response(ERR)
                        ERR→sump-sender
                        respond RELEASEind(REQ,ERR)
                        set IDLE
```

```
                    }|
                error perr:   // Parsing Error
                    err_code = get_parse_err_code_from_matrix(perr)
                    err_msg = get_ parse_err_msg_from_matrix(err_code)
                    respond ERRORind(REQ, err_code)
                    set ERROR
                    var ERR: RES
                    ERR.status.code = err_code
                    ERR.status.msg = err_msg
                    coding_response(ERR)
                    ERR→sump-sender
                    respond RELEASEind(REQ,ERR)
                    set IDLE
            }
    end|
(req_too_long or too_many_reqs or header_too_large or bad_header) err:
    begin
            err_code = get_recv_err_code_from_matrix(err)
            err_msg = get_ recv_err_msg_from_matrix(err_code)
            respond ERRORind(REQ, err_code)
            set ERROR
            var ERR: RES
            ERR.status.code = err_code
            ERR.status.msg = err_msg
            coding_response(ERR)
            ERR→sump-sender
            respond RELEASEind(REQ,ERR)
            set IDLE
    end
} // wait-event main


protocol part response_modu (peer sump- receiver)
var err_code: integer init(0)
var err, err_msg: [ ] byte
var err_matrix: array[15] of record(sign: [ ] byte, code: integer, msg: [ ] byte)
err_matrix = {
    {“req_too_long”,413,”REQUEST TOO LONG”},
    {“too_many_reqs”,429,”TOO MANY REQUEST”},
    {“heaer_too_large”,431,”HEADER TOO LARGE”},
    {“bad_header”,432,”BAD HEADER”},
    {“bad_request”,400,”BAD REQUEST”},
    {“method_not_allow”,405,”METHOD NOT ALLOW”},
    {“format_unsupported”,406,”FORMAT NOT SUPPORTED”},
    {“not_implemented”,501,”NOT IMPLEMENTED”},
    {“ver_not_supported”,505,”VERSION NOT SUPPORTED”},
    {“unauthorized”,401,”UNAUTHORIZED”},
```

```
    {“forbidden”,403,”FORBIDDEN”},{“not_found”,404,”NOT FOUND”},
    {“internal”,500,”INTERNAL ERROR”},{“unavailable”,503,”UNAVAILABLE”},
    {“timeout”,504,”TIMEOUT”},{“disk_full”,507,”NOT ENOUGH SPACE”}
}
timer t: 0..? ms                                        // Cronometro
set RECEIVING REQUEST
wait event {
REQ ← sump-sender and                                   // Trigger Event
    (REQ.method=“GET_MODULE” or REQ.method=“GET_LIST_MODULE”):
    begin
            set PARSER
            wait event{
              request_ok:
                    set PROCESS REQUEST
                    start t
                    if (REQ.method = “GET_MODULE”)
                            respond MODULEind(REQ)
                    if (REQ.method = “GET_LIST_MODULE”)
                            respond LIST_MODULEind(REQ)
                    wait event{
                      RES ← sump-receiver:
                            coding_response(RES)
                            RES→sump-sender
                            reset t
                            set FIN
                            respond RELEASEind(REQ,RES)
                            set IDLE |
                      error err:     // Processing Error
                            reset t
                            err_code = get_proc_err_code_from_matrix(err)
                            err_msg = get_ proc_err_msg_from_matrix(err_code)
                            respond ERRORind(REQ, err_code)
                            set ERROR
                            var ERR: RES
                            ERR.status.code = err_code
                            ERR.status.msg = err_msg
                            coding_response(ERR)
                            ERR→sump-sender
                            respond RELEASEind(REQ,ERR)
                            set IDLE
                    }|
              error perr:   // Parsing Error
                    err_code = get_parse_err_code_from_matrix(perr)
                    err_msg = get_ parse_err_msg_from_matrix(err_code)
                    respond ERRORind(REQ, err_code)
                    set ERROR
                    var ERR: RES
```

```
                    ERR.status.code = err_code
                    ERR.status.msg = err_msg
                    coding_response(ERR)
                    ERR→sump-sender
                    respond RELEASEind(REQ,ERR)
                    set IDLE
            }
    end|
(req_too_long or too_many_reqs or header_too_large or bad_header) err:
    begin
            err_code = get_recv_err_code_from_matrix(err)
            err_msg = get_ recv_err_msg_from_matrix(err_code)
            respond ERRORind(REQ, err_code)
            set ERROR
            var ERR: RES
            ERR.status.code = err_code
            ERR.status.msg = err_msg
            coding_response(ERR)
            ERR→sump-sender
            respond RELEASEind(REQ,ERR)
            set IDLE
    end
} // wait-event main

protocol part response_policy (peer sump- receiver)
var err_code: integer init(0)
var err, err_msg: [ ] byte
var err_matrix: array[15] of record(sign: [ ] byte, code: integer, msg: [ ] byte)
err_matrix = {
    {"req_too_long",413,"REQUEST TOO LONG"},
    {"too_many_reqs",429,"TOO MANY REQUEST"},
    {"heaer_too_large",431,"HEADER TOO LARGE"},
    {"bad_header",432,"BAD HEADER"},
    {"bad_request",400,"BAD REQUEST"},
    {"method_not_allow",405,"METHOD NOT ALLOW"},
    {"format_unsupported",406,"FORMAT NOT SUPPORTED"},
    {"not_implemented",501,"NOT IMPLEMENTED"},
    {"ver_not_supported",505,"VERSION NOT SUPPORTED"},
    {"unauthorized",401,"UNAUTHORIZED"},
    {"forbidden",403,"FORBIDDEN"},{"not_found",404,"NOT FOUND"},
    {"internal",500,"INTERNAL ERROR"},{"unavailable",503,"UNAVAILABLE"},
    {"timeout",504,"TIMEOUT"},{"disk_full",507,"NOT ENOUGH SPACE"}
}
timer t: 0..? ms                                    // Cronometro
set RECEIVING REQUEST
wait event {
REQ ← sump-sender and                               // Trigger Event
```

```
(REQ.method = “GET_POLICY” or REQ.method = “GET_LIST_POLICY”):
begin
        set PARSER
        wait event{
          request_ok:
                set PROCESS REQUEST
                start t
                if (REQ.method = “GET_POLICY”)
                        respond POLICYind(REQ)
                if (REQ.method = “GET_LIST_POLICY”)
                        respond LIST_POLICYind(REQ)
                wait event{
                  RES ← sump-receiver:
                        coding_response(RES)
                        RES→sump-sender
                        reset t
                        set FIN
                        respond RELEASEind(REQ,RES)
                        set IDLE |
                  error err:     // Processing Error
                        reset t
                        err_code = get_proc_err_code_from_matrix(err)
                        err_msg = get_ proc_err_msg_from_matrix(err_code)
                        respond ERRORind(REQ, err_code)
                        set ERROR
                        var ERR: RES
                        ERR.status.code = err_code
                        ERR.status.msg = err_msg
                        coding_response(ERR)
                        ERR→sump-sender
                        respond RELEASEind(REQ,ERR)
                        set IDLE
                }|
            error perr:   // Parsing Error
                err_code = get_parse_err_code_from_matrix(perr)
                err_msg = get_ parse_err_msg_from_matrix(err_code)
                respond ERRORind(REQ, err_code)
                set ERROR
                var ERR: RES
                ERR.status.code = err_code
                ERR.status.msg = err_msg
                coding_response(ERR)
                ERR→sump-sender
                respond RELEASEind(REQ,ERR)
                set IDLE
        }
end|
```

```
(req_too_long or too_many_reqs or header_too_large or bad_header) err:
   begin
         err_code = get_recv_err_code_from_matrix(err)
         err_msg = get_ recv_err_msg_from_matrix(err_code)
         respond ERRORind(REQ, err_code)
         set ERROR
         var ERR: RES
         ERR.status.code = err_code
         ERR.status.msg = err_msg
         coding_response(ERR)
         ERR→sump-sender
         respond RELEASEind(REQ,ERR)
         set IDLE
   end
} // wait-event main

protocol part response_info (peer sump- receiver)
var err_code: integer init(0)
var err, err_msg: [ ] byte
var err_matrix: array[15] of record(sign: [ ] byte, code: integer, msg: [ ] byte)
err_matrix = {
   {“req_too_long”,413,”REQUEST TOO LONG”},
   {“too_many_reqs”,429,”TOO MANY REQUEST”},
   {“heaer_too_large”,431,”HEADER TOO LARGE”},
   {“bad_header”,432,”BAD HEADER”},
   {“bad_request”,400,”BAD REQUEST”},
   {“method_not_allow”,405,”METHOD NOT ALLOW”},
   {“format_unsupported”,406,”FORMAT NOT SUPPORTED”},
   {“not_implemented”,501,”NOT IMPLEMENTED”},
   {“ver_not_supported”,505,”VERSION NOT SUPPORTED”},
   {“unauthorized”,401,”UNAUTHORIZED”},
   {“forbidden”,403,”FORBIDDEN”},{“not_found”,404,”NOT FOUND”},
   {“internal”,500,”INTERNAL ERROR”},{“unavailable”,503,”UNAVAILABLE”},
   {“timeout”,504,”TIMEOUT”},{“disk_full”,507,”NOT ENOUGH SPACE”}
}
timer t: 0..? ms                                    // Cronometro
set RECEIVING REQUEST
wait event {
REQ ← sump-sender and                               // Trigger Event
   (REQ.method = “SET_INFO”):
   begin
         set PARSER
         wait event{
           request_ok:
                set PROCESS REQUEST
                start t
                if (REQ.method = “SET_INFO”)
```

```
                    respond SET_INFOind(REQ)
                wait event{
                  RES ← sump-receiver:
                    coding_response(RES)
                    RES→sump-sender
                    reset t
                    set FIN
                    respond RELEASEind(REQ,RES)
                    set IDLE |
                  error err:     // Processing Error
                    reset t
                    err_code = get_proc_err_code_from_matrix(err)
                    err_msg = get_ proc_err_msg_from_matrix(err_code)
                    respond ERRORind(REQ, err_code)
                    set ERROR
                    var ERR: RES
                    ERR.status.code = err_code
                    ERR.status.msg = err_msg
                    coding_response(ERR)
                    ERR→sump-sender
                    respond RELEASEind(REQ,ERR)
                    set IDLE
                }|
              error perr:   // Parsing Error
                err_code = get_parse_err_code_from_matrix(perr)
                err_msg = get_ parse_err_msg_from_matrix(err_code)
                respond ERRORind(REQ, err_code)
                set ERROR
                var ERR: RES
                ERR.status.code = err_code
                ERR.status.msg = err_msg
                coding_response(ERR)
                ERR→sump-sender
                respond RELEASEind(REQ,ERR)
                set IDLE
            }
    end|
(req_too_long or too_many_reqs or header_too_large or bad_header) err:
    begin
            err_code = get_recv_err_code_from_matrix(err)
            err_msg = get_ recv_err_msg_from_matrix(err_code)
            respond ERRORind(REQ, err_code)
            set ERROR
            var ERR: RES
            ERR.status.code = err_code
            ERR.status.msg = err_msg
            coding_response(ERR)
```

```
            ERR→sump-sender
            respond RELEASEind(REQ,ERR)
            set IDLE
        end
    } // wait-event main


    } // Fin de especificación SUMP
```

4.1.3.1.3. Propuesta de Especificación de Implementación

Esta propuesta describe el prototipo de implementación del protocolo SUMP en un entorno de ejecución controlado por el investigador. Se siguen los 4 pasos sugeridos por König para esta actividad, por lo tanto, no se contemplan técnicas de descripción formal y en su lugar, el investigador propone en cada paso la información técnica relevante para la posterior construcción del protocolo.

4.1.3.1.3.1. Analisis de restricciones

En el entorno de ejecución de la propuesta de implementación, se tienen algunas características y restricciones que el autor considera oportuno indicar para tener un mejor panorama durante la construcción del simulador.

Tabla 29
Características del entorno de ejecución

Característica	Descripción
Rápido	La implementación debe prestar una especial atención a la rapidez en la ejecución. Se debe preferir la rapidez en lugar de muchos requerimientos no funcionales que puedan retrasar su respuesta.
Simple	La implementación debe ser simple de instalar y utilizar. Debe evitarse la instalación de software adicional para poder utilizarlo, caso contrario debe proporcionar los componentes adicionales como parte del software de tal forma que esta simplicidad se cumpla. El uso del software también debe ser intuitivo al usuario y su configuración auto explicativa.
Seguro	La información que procesa el protocolo es crítica por lo que se deben tomar las medidas necesarias para asegurarla. Toda información que pase a través del protocolo debe cifrarse, sin embargo no es estrictamente necesario que los procesos de cifrado se lleven a cabo dentro de los mecanismos del protocolo. Por lo tanto, es recomendable que el cifrado lo lleve a cabo otro sistema externo a la

	implementación (p.e. ssh tunnel).
Modular	La implementación debe tener presente un diseño modular que permita extender la funcionalidad existente sin afectar la lógica principal del protocolo.

Fuente: Elaboración del investigador

Tabla 30
Restricciones del entorno de ejecución

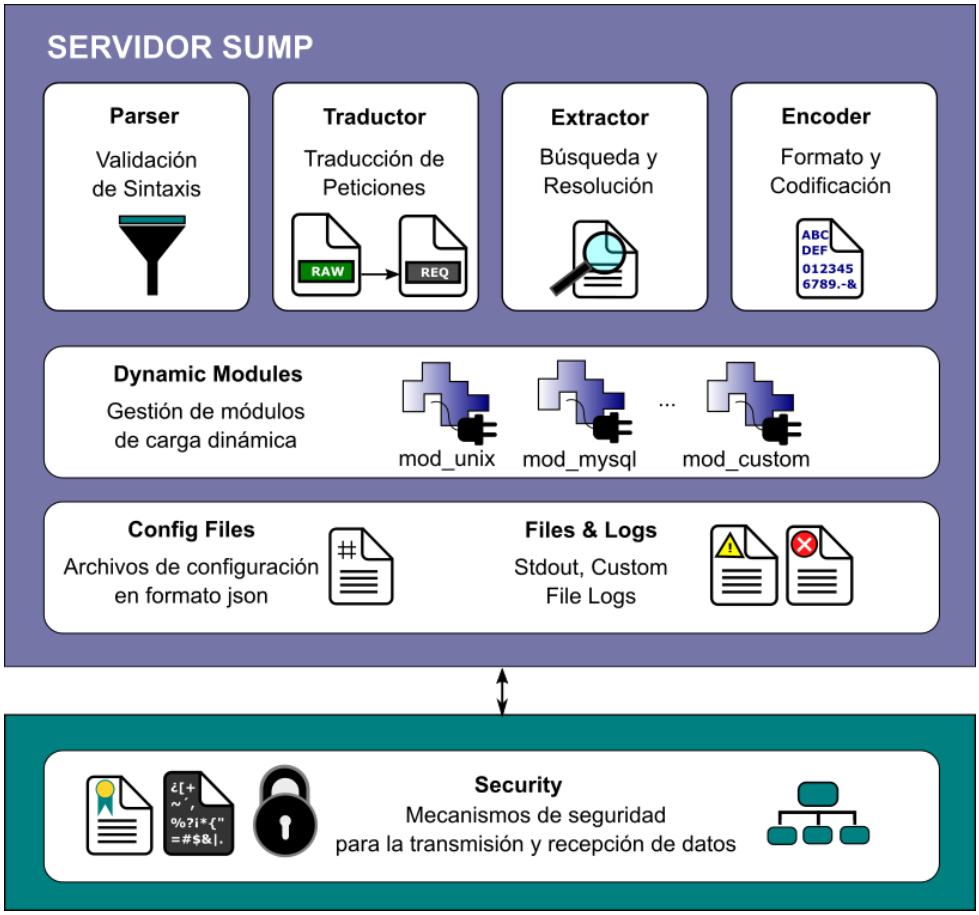
Restricción	Descripción
Sistema Operativo	El sistema operativo debe ser Linux, concretamente la versión Red Hat Enterprise Linux 6.6 debido a que la rama 6.0 de este sistema operativo es el de mayor uso en la sede de estudio.
Lenguaje de Programación	El lenguaje de programación por excelencia para la implementación de protocolos es el lenguaje C, sin embargo, también pueden utilizarse otros lenguajes como Go u otros menos potentes pero más sencillos como Python.
Librerías existentes	Entre las librerías existentes a utilizar debe contemplarse al menos las siguientes en el caso de utilizar el lenguaje C: mysql-devel, pwd.h, shadow.h, pthreads, dl. En caso se utilice el lenguaje Go: plugin, encondig/json, go-sql-driver/mysql, os/user. En caso se use otro lenguaje de programación, se deben buscar librerías similares o envoltorios hacia estas.
Plataforma de simulación	El funcionamiento del protocolo debe comprobarse en un entorno virtualizado que simule todas las características técnicas del sistema operativo redhat y la red de servidores de la sede de estudio. Se recomienda utilizar openvz sobre VMware ESX 5.0 con una plantilla para el sistema operativo RHEL 6.0.

Fuente: Elaboración del investigador

4.1.3.1.3.2. Implementación local relevante

La propuesta de diseño de implementación contempla decisiones técnicas mucho más precisas, representadas en una arquitectura de componentes que el autor considera muy importante indicar. Esta arquitectura muestra la relación de los componentes del software que será utilizado para brindar el soporte necesario al protocolo SUMP en el lado del servidor y del cliente.

Gráfico 19: Arquitectura del servidor SUMP



Fuente: Elaboración del investigador

Los componentes que pueden apreciar, se detallan a continuación

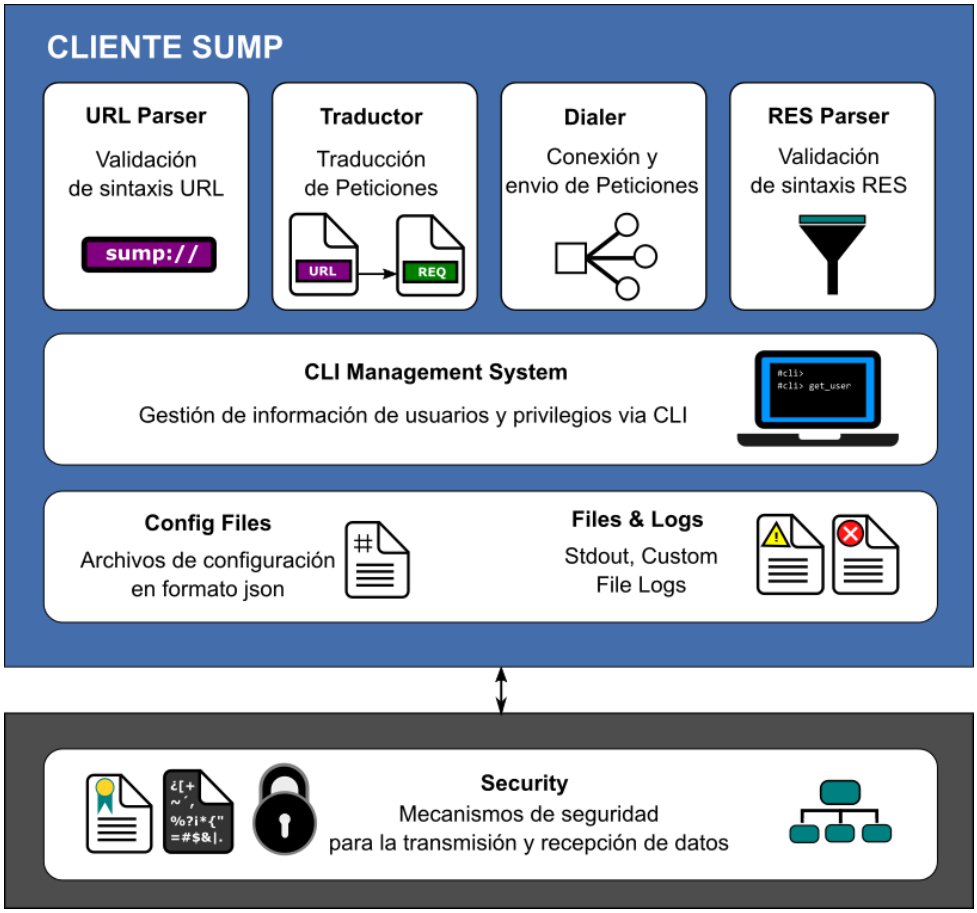
Tabla 31
Componentes de la arquitectura del servidor SUMP

Componente	Descripción
Parser	Permite validar las peticiones para que cumplan con toda la sintaxis del protocolo
Traductor	Permite traducir la petición del formato RAW en una estructura REQUEST
Extractor	Permite buscar y resolver en los módulos dinámicos la información solicitada.
Encoder	Permite aplicar formato y codificación a la estructura RESPONSE para ser enviada al cliente
Config Files	Permite trabajar con un archivo de configuración en formato json donde se guarda información relevante para el comportamiento del servidor.
Dynamic Modules	Son los módulos que darán soporte a las funciones de extracción de datos.
Files & Logs	Facilita funcionalidad básica para operar con

	archivos y logs que permitan hacer seguimiento de auditoría ante problemas.
Security	Provee de mecanismos de protección de datos en las comunicaciones del servidor. Este componente puede estar incluido dentro del protocolo. Sin embargo, al ser un tópico complejo, se recomienda que este componente pueda delegarse a otras soluciones más robustas y probadas como túneles SSH, VPN, Wrappers TLS, etc.

Fuente: Elaboración del investigador

Gráfico 20: Arquitectura del cliente SUMP



Fuente: Elaboración del investigador

Los componentes para el prototipo del lado del cliente, se detallan a continuación

Tabla 32
Componentes de la arquitectura del cliente SUMP

Componente	Descripción
URL Parser	Permite validar la sintaxis del URL para el protocolo

	SUMP
Traductor	Permite traducir la URL en una estructura REQUEST
Dialer	Permite establecer comunicación con el servidor SUMP y enviar las peticiones
RES Parser	Permite validar la sintaxis de la respuesta del servidor ante una petición y enviar los resultados al CLI Management System.
CLI Management System	Permite gestionar y procesar la información de los usuarios y sus privilegios mediante el envío de URL y la recepción de información desde el RES Parser. Es el orquestador de los componentes del cliente.
Config Files	Permite trabajar con un archivo de configuración en formato json donde se guarda información relevante para el comportamiento del cliente.
Files & Logs	Facilita funcionalidad básica para operar con archivos y logs que permitan hacer seguimiento de auditoría ante problemas.
Security	Provee de mecanismos de protección de datos en las comunicaciones del servidor. Este componente puede estar incluido dentro del cliente. Sin embargo, al ser un tópico complejo, se recomienda que este componente pueda delegarse a otras soluciones más robustas y probadas como túneles SSH, VPN, Wrappers TLS, etc.

Fuente: Elaboración del investigador

4.1.3.1.3.3. Elección del modelo de implementación

El modelo de implementación elegido es el modelo servidor, y el mapeo de los procesos asociados a este servidor es el siguiente.

Tabla 33
Mapeo de procesos del servidor SUMP

Evento	**Respuesta del proceso**
GET_USER	Busca y devuelve información del usuario en el subsistema (modulo) especificado. Si no encuentra la información u ocurre algún error durante el proceso, devuelve información de diagnóstico para su revisión.
GET_LIST_USER	Busca y devuelve información de la lista de usuarios en el subsistema (modulo) especificado. Si no encuentra la información u ocurre algún error durante el proceso, devuelve información de diagnóstico para su revisión.
GET_GROUP	Busca y devuelve información del grupo en el subsistema (modulo) especificado. Si no encuentra la información u ocurre algún error durante el proceso, devuelve información de diagnóstico para

	su revisión.
GET_LIST_GROUP	Busca y devuelve información de la lista de grupos en el subsistema (modulo) especificado. Si no encuentra la información u ocurre algún error durante el proceso, devuelve información de diagnóstico para su revisión.
GET_MODULE	Busca y devuelve información del módulo especificado. Si no encuentra la información u ocurre algún error durante el proceso, devuelve información de diagnóstico para su revisión.
GET_LIST_MODULE	Busca y devuelve información de la lista de módulos soportado por el servidor. Si no encuentra la información u ocurre algún error durante el proceso, devuelve información de diagnóstico para su revisión.
GET_POLICY	Busca y devuelve información de la política que se aplica en el subsistema (modulo) especificado para un usuario o grupo determinado. Si no encuentra la información u ocurre algún error durante el proceso, devuelve información de diagnóstico para su revisión.
GET_LIST_POLICY	Busca y devuelve información de las políticas que se aplican en el subsistema (modulo) especificado para un usuario o grupo determinado. Si no encuentra la información u ocurre algún error durante el proceso, devuelve información de diagnóstico para su revisión.
SET_INFO	Manipula información de los usuarios, grupos y políticas que se aplican en el subsistema (módulo) especificado. Si no encuentra la información a manipular u ocurre algún error durante el proceso, devuelve información de diagnóstico para su revisión.

Fuente: Elaboración del investigador

4.1.3.1.3.4. Especificación de la implementación

Finalmente, el investigador ha creído conveniente que la especificación de la implementación del protocolo SUMP debe ser representada mediante el formato RPC, para ser mejor entendida, debido a la naturaleza técnica de esta especificación.

Esto en razón de que un documento RFC puede incluir en un mismo documento, los detalles técnicos precisos y delimitados que un implementador de software necesita para poder construir el protocolo en el lenguaje de programación que considere pertinente. Este documento RFC,

al tener un formato de regular extensión, ha sido incluido en el **Anexo Nro 15** de esta investigación.

4.1.3.1.4. Simulación del protocolo SUMP

Aplicación del simulador del protocolo SUMP para la transmisión de datos de cuentas de usuarios y privilegios en un entorno virtualizado de la empresa Petroperú en su sede Iquitos.

Para cumplir este procedimiento se realizaron las siguientes actividades

Tabla 34
Actividades para cumplir el objetivo 3

Actividad	Descripción
Programación del protocolo sump	Siguiendo el ciclo de desarrollo de software, se construye un prototipo en lenguaje Go del protocolo propuesto. Este software es llamado SUMP (Simple User Management Protocol)
Instalación y Configuración del entorno virtualizado	Instalar y configurar un servidor OpenVZ dentro de un servidor VMware ESX para proporcionar la infraestructura de pruebas
Creación de Máquinas Virtuales	Crear las máquinas virtuales correspondientes a los servicios incluidos dentro de la tabla 18
Pruebas del protocolo	Realizar las pruebas del nuevo protocolo implementado en el entorno virtualizado

Fuente: Elaboración del investigador

a) Programación del protocolo SUMP

Para el desarrollo de esta actividad se siguió un ciclo de desarrollo personalizado conformado por las siguientes fases: Análisis, Diseño, Codificación y Pruebas.

La programación se llevó a cabo en lenguaje Go debido a que se requiere una implementación robusta, nativa y eficiente. El lenguaje Go cumple con estas tres condiciones y brinda además el beneficio de la simplicidad de su sintaxis.

El programa sump contiene las siguientes características generales

- ❖ Tiene un diseño modular (vía el paquete plugin)
- ❖ Usa archivos de configuración para controlar su comportamiento
- ❖ Soporta los módulos Linux y Mysql
- ❖ La información se manipula directamente desde la memoria

b) Instalación y Configuración del entorno virtualizado

Se eligió utilizar un entorno virtualizado para evitar algún problema real en la red a causa de un procedimiento mal realizado durante la fase de codificación y pruebas. La infraestructura utilizada se puede apreciar en el **Anexo 05.**

Se utilizó un servidor VMware ESX de pruebas para construir el entorno virtualizado. Se eligió la tecnología VMware debido a que es la que se utiliza en Petroperú sede Iquitos.

Debido a las limitaciones de espacio en disco y memoria, se utilizó OpenVZ para simular toda la infraestructura de servidores seleccionada para las pruebas del protocolo (**Ver Anexo 06**).

c) Creación de Máquinas Virtuales

Se utilizaron imágenes de los sistemas operativos Red Hat Enterprise Linux 6 para la creación de máquinas virtuales. La configuración de red, hostname, usuarios y servicios fue alterada para evitar entregar información sensible, de acuerdo a las normas internas de la empresa (**Ver Anexo 07 y 08**).

d) Pruebas del protocolo

Las pruebas del protocolo se realizaron con el software cliente a través de un túnel encriptado mediante el protocolo SSH. El protocolo SSH que viene incluido por defecto en los servidores Red Hat mediante el software OpenSSL.

La arquitectura utilizada para la simulación del protocolo SUMP se puede apreciar en el **Anexo 09.**

Durante las pruebas, se utilizaron los certificados digitales integrados en el servicio SSH y se configuró el nivel de encriptación a 4906 bits.

Según los resultados obtenidos en el Objetivo Nro 3, se evidencia que, el diseño del protocolo SUMP requirió de la Elaboración de la Propuesta de Protocolo, la cual consiste en la puesta en práctica de la metodología mixta propuesta por el investigador en la dimensión metodológica de diseño, contemplando sus 4 fases. En esta propuesta se puede apreciar la influencia de König en el diseño del protocolo al abordar sus 3 fases de especificaciones.

Durante la fase de simulación, el diseño del protocolo SUMP requirió de la ejecución de 4 actividades relacionadas: Programación del protocolo Sump, Instalación y Configuración del entorno virtualizado, Creación de Máquinas Virtuales y Pruebas del protocolo.

La programación del protocolo SUMP en lenguaje Go se realizó siguiendo algunas recomendaciones emitidas por Hartmut König para el diseño y desarrollo de protocolos de comunicación. El resultado de la programación se concreta en el servidor y cliente sump cuya arquitectura puede apreciarse en el **Gráficos 19 y 20.**

El código fuente puede revisarse en el DVD del proyecto de Tesis, en la carpeta software\sump.

La Instalación y Configuración del entorno virtualizado, Creación de Máquinas Virtuales y Pruebas del protocolo se efectuaron de forma exitosa según lo indicado en sus anexos respectivos (Anexo 05, 06, 07, 08, 09, 10, 11 y 12).

Por consiguiente, al haber presentado todos los entregables necesarios según la metodología mixta aplicada en esta investigación, los resultados consiguen la categoría de **OPTIMO** para este objetivo.

4.1.4. Objetivo 4

Evaluar la simulación del protocolo SUMP en la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en la sede Iquitos.

Relativo a la Performance

A los sistemas y servicios que conforman el grupo de estudio se les aplicó el Post test, con el propósito de determinar el tiempo en minutos (Xi) que toma atender requerimientos con la simulación del protocolo SUMP para atender requerimientos del proceso actual de monitoreo de las cuentas de usuarios y sus privilegios y los resultados fueron los siguientes.

Tabla 35
Post-Test – Frecuencia de tiempos de respuesta al grupo experimental

Grupo Experimental		
Xi	**Frecuencia**	**Porcentaje**
1	11	100.000%
3	0	0.000%
5	0	0.000%
10	0	0.000%
Total	**11**	**100.000%**

Fuente: Post-Test Petroperú-Iquitos

Fecha: 2015-07-03

Luego, tras contrastar los resultados previos con el Cuadro de categorización de performance para dimensión metodológica de Evaluación, se obtuvieron las siguientes categorías de tiempo de respuesta:

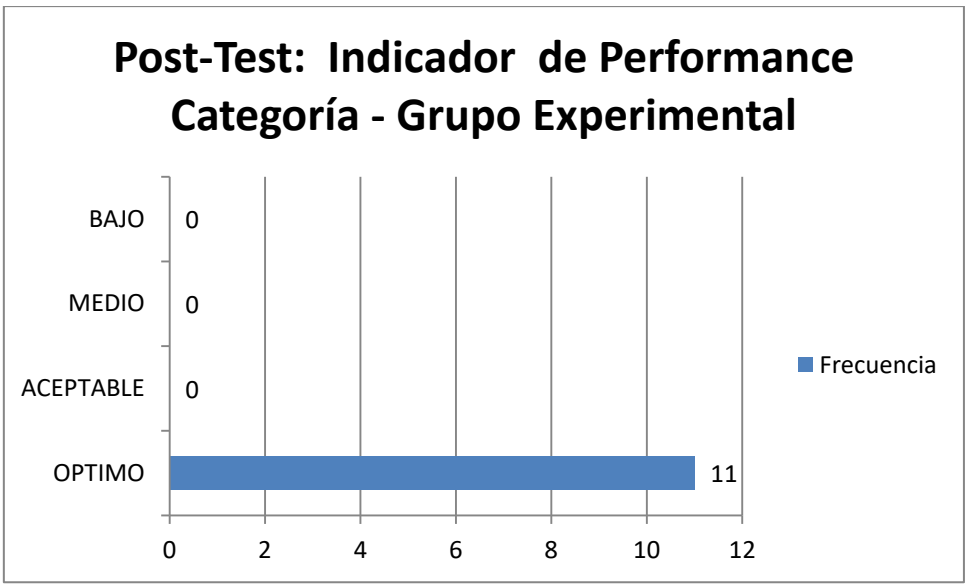
Tabla 36
Post-Test - Categoría de tiempos de respuesta al grupo experimental

Categoría - Grupo Experimental		
Categoría	**Frecuencia**	**Porcentaje**
OPTIMO	11	100.000%
ACEPTABLE	0	0.000%
MEDIO	0	0.000%
BAJO	0	0.000%
Total	**11**	**100.000%**

Fuente: Tabla 35

Fecha: 2015-07-03

Gráfico 21: Post-Test - Categoría de tiempos de respuesta al grupo experimental



Fuente: Tabla 35

Fecha: 2015-07-03

Los resultados obtenidos del post-test por categorías para medir el tiempo de respuesta en minutos (Xi) que toma atender requerimientos con el protocolo SUMP para el monitoreo de las cuentas de usuarios y sus privilegios fueron los siguientes:

En la categoría **BAJO** no se encontraron sistemas y servicios que requieran la atención de sus requerimientos con tiempos de respuesta entre 10 a más minutos. Esto representa el 0% de los resultados.

En la categoría **MEDIO** no se encontraron sistemas y servicios que requieran la atención de sus requerimientos con tiempos de respuesta entre 4 a 9 minutos. Esto representa el 0% de los resultados.

En la categoría **ACEPTABLE** no se encontraron sistemas y servicios que requieran la atención de sus requerimientos con tiempos de respuesta entre 2 a 3 minutos. Esto representa el 0% de los resultados.

En la categoría **OPTIMO** se encontraron 11 sistemas y servicios que requieran la atención de sus requerimientos con tiempos de respuesta entre 0 y 1 minutos. Esto representa el 100% de los resultados. Este indicador demuestra que el proceso de revisión mediante el protocolo SUMP mejora significativamente este procedimiento.

Tabla 37
Post-Test - Resultados del Post-Test al grupo experimental

Resultado Post-Test – Indicador de Performance – Grupo Experimental			**Estadígrafos**
Categoría	**Frecuencia**	**Porcentaje**	
OPTIMO	11	100.000%	Media: 1 Desv: 0 Cv: 0%
ACEPTABLE	0	0.000%	
MEDIO	0	0.000%	
BAJO	0	0.000%	
Total	**11**	**100.000%**	

Fuente: Tabla 36

Fecha: 2015-07-03

El valor de la media aritmética obtenido por los sistemas y servicios del grupo experimental en el pre-test para medir el tiempo de respuesta en minutos (Xi) que toma atender requerimientos según el proceso actual de monitoreo de las cuentas de usuarios y sus privilegios es de **1**, lo cual nos indica que se encuentra dentro de la categoría **OPTIMO** dentro de la escala de la variable dependiente evaluada.

El valor de la desviación estándar es de **0** nos indica que la mayoría de valores se distribuyen a esa distancia a la derecha e izquierda alrededor del promedio.

Finalmente, se observa un coeficiente de variación de **0%** lo cual nos indica un resultado homogéneo (menor a 32%).

Relativo a la Seguridad

A los sistemas y servicios que conforman el grupo de estudio se les aplicó el Post-Test, con el propósito de determinar cuan seguro es atender requerimientos del proceso de monitoreo de las cuentas de usuarios y sus privilegios, siguiendo las

recomendaciones establecidas con el protocolo SUMP y los resultados fueron los siguientes.

Tabla 38
Post-Test – Indicador Seguridad - Resultados del Post-Test al grupo experimental

Servidor	Sistema o Servicio	Autenticación	Control de Conexiones	Encriptación	Categoría
Proxy Server	Linux	SI	SI	4096bits	Optimo
Mail Server	Linux	SI	SI	4096bits	Optimo
SMB Server	Linux	SI	SI	4096bits	Optimo
Central Server	Linux	SI	SI	4096bits	Optimo
Central Server	MySQL	SI	SI	4096bits	Optimo
Backup Server	Linux	SI	SI	4096bits	Optimo
Syslog Server	Linux	SI	SI	4096bits	Optimo
Monitor Server 1	Linux	SI	SI	4096bits	Optimo
Monitor Server 1	MySQL	SI	SI	4096bits	Optimo
Monitor Server 2	MySQL	SI	SI	4096bits	Optimo
Directory Server	Linux	SI	SI	4096bits	Optimo

Fuente: Post-Test Petroperú-Iquitos

Fecha: 2015-07-03

Luego, tras contrastar los resultados previos con el Cuadro de categorización de seguridad para dimensión metodológica de Evaluación, se obtuvieron las siguientes categorías:

Tabla 39
Post-Test – Indicador Seguridad - Categoría al grupo experimental

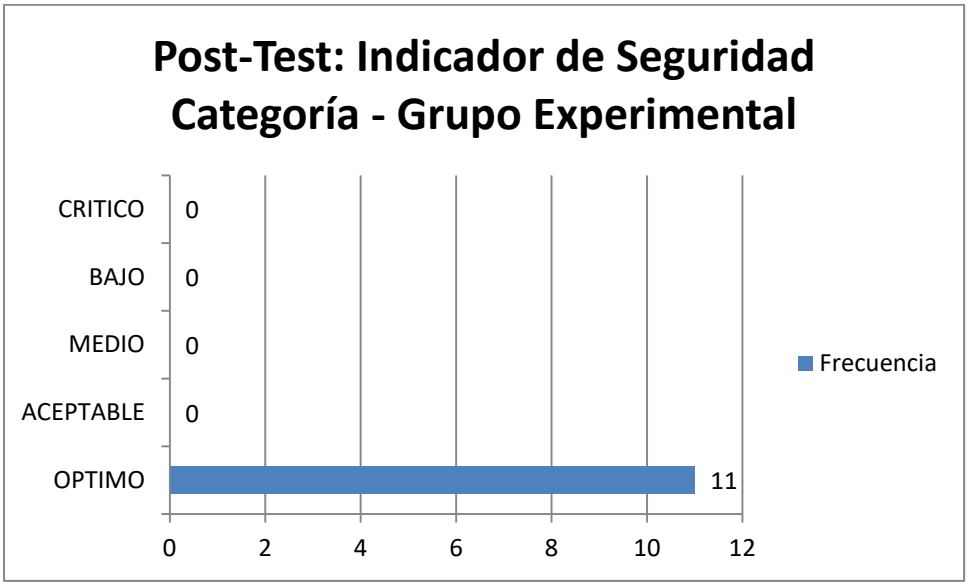
Categoría - Grupo Experimental		
Categoría	**Frecuencia**	**Porcentaje**
OPTIMO	11	100.000%
ACEPTABLE	0	0.000%
MEDIO	0	0.000%
BAJO	0	0.000%

CRITICO	0	0.000%
Total	**11**	**100.000%**

Fuente: Tabla 38

Fecha: 2015-07-03

Gráfico 22: Post-Test – Indicador Seguridad - Categoría al grupo experimental



Fuente: Tabla 38

Fecha: 2015-07-03

Los resultados mostrados en el gráfico, evidencian de manera clara que 11 sistemas y servicios tienen la categoría OPTIMO, es decir todos los sistemas y servicios. Para resumir los resultados obtenidos se elaboró el siguiente cuadro:

Tabla 40
Post-Test – Indicador de Seguridad - Resultados del Post-Test al grupo experimental

Resultado Post-Test – Indicador Seguridad – Grupo Experimental		
Categoría	**Frecuencia**	**Porcentaje**
OPTIMO	11	100.000%
ACEPTABLE	0	0.000%
MEDIO	0	0.000%
BAJO	0	0.000%
CRITICO	0	0.000%
Total	**11**	**100.000%**

Fuente: Tabla 38

Fecha: 2015-07-03

Los resultados obtenidos por categorías para medir el nivel de seguridad para atender requerimientos según el proceso actual de monitoreo de las cuentas de usuarios y sus privilegios fueron los siguientes:

En la categoría **CRÍTICO** no se encontraron sistemas y servicios que no usen mecanismo de autenticación, que no usen controles de conexión y que no cifren sus comunicaciones. Esto representa el 0% de los resultados.

En la categoría **BAJO** no se encontraron sistemas y servicios que si usen mecanismos de autenticación, pero que no cuenten con controles de conexión y cuyo cifrado sea menor o igual a 512 bits en sus comunicaciones. Esto representa el 0% de los resultados.

En la categoría **MEDIO** no se encontraron sistemas y servicios que si utilizan mecanismos de autenticación, no cuentan con un control de conexiones y utilizan una encriptación mayor o igual a 1024bits y menor o igual a 2048bits. Esto representa el 0% de los resultados.

En la categoría **ACEPTABLE** no se encontraron sistemas y servicios que si utilizan mecanismos de autenticación, si cuentan con un control de conexiones y utilizan una encriptación mayor o igual a 1024bits y menor o igual a 2048bits. Esto representa el 0% de los resultados.

En la categoría **ÓPTIMO** se encontraron 11 sistemas y servicios que utilizan un mecanismo de autenticación, cuentan con controles de conexión, y utilizan una encriptación mayor o igual a 4096bits. Esto representa el 100% de los resultados. Este indicador demuestra que el proceso de revisión mejora notablemente con el protocolo SUMP.

Comparación de los resultados obtenidos del pre-test y post-test para determinar el nivel de mejora en la revisión de los derechos de acceso de los usuarios.

Relativo a la Performance

Al analizar los resultados obtenidos en el pre-test y post-test al grupo experimental se pudo comparar el tiempo de respuesta en minutos que toma atender requerimientos de monitoreo de las cuentas de usuarios y sus privilegios. Los resultados fueron los siguientes:

Tabla 41
Resultados comparativos relativos al Indicador de Performance por categoría entre el Pre-Test y Post-Test al Grupo Experimental

Categoría	Pre-Test G.E.	Post-Test G.E.
OPTIMO	0.000%	100.000%
ACEPTABLE	72.727%	0.000%
MEDIO	27.273%	0.000%
BAJO	0.000%	0.000%

Fuente: Resultados del Pre-Test y Post-Test relativos a la performance

Fecha: 2015-07-03

En la tabla se puede apreciar, que tras aplicar el estímulo al grupo experimental ocurre una mejora significativa en la performance referente al tiempo de respuesta en minutos que toma atender requerimientos de monitoreo de las cuentas de usuarios y sus privilegios. Esto se sustenta en que el **100% de los sistemas y servicios consiguieron la categoría OPTIMO** en el post-test, mientras que en el pre-test los valores encontrados fueron del 72.727% (ACEPTABLE), un 27.273% (MEDIO) y 0% (BAJO) respectivamente.

Tabla 42
Índices estadísticos comparativos relativos al Indicador de Performance por categoría entre el Pre-Test y Post-Test al Grupo Experimental

Índice	Pre-Test G.E.	Post-Test G.E.
Muestra	11	11
Media Aritmética	3.5454	1
Desviación Estándar	0.8727	0

Coeficiente de Variación	24.6153%	0%

Fuente: Tabla 41

Fecha: 2015-07-03

El valor de la media aritmética obtenido por los sistemas y servicios del grupo experimental en el pre-test para medir el tiempo de respuesta en minutos (Xi) que toma atender requerimientos según el proceso actual de monitoreo de las cuentas de usuarios y sus privilegios es de **3.5454** en el Pre-Test y **1** en el Post-Test, lo cual nos indica que se mejoró desde la categoría **ACEPTABLE** en el Pres-Test a la categoría **OPTIMO** en el Post-Test dentro de la escala de la variable dependiente evaluada.

El valor de la desviación estándar es de **0.8727** para el Pre-Test y **0** para el Post-Test lo cual nos indica que la mayoría de valores se distribuyen a esa distancia a la derecha e izquierda alrededor del promedio para el Pre-Test y que la desviación fue de cero en el Post-Test.

Finalmente, se observa un coeficiente de variación fue de **24.6153%** para el Pre-Test y **0** para el Post-Test, lo cual nos indica un resultado homogéneo.

Relativo a la Seguridad

Al analizar los resultados obtenidos en el pre-test y post-test al grupo experimental se pudo comparar el nivel de seguridad durante la atención de los requerimientos de monitoreo de las cuentas de usuarios y sus privilegios. Los resultados fueron los siguientes:

Tabla 43
Resultados comparativos relativos al Indicador de Seguridad por categoría entre el Pre-Test y Post-Test al Grupo Experimental

Categoría	Pre-Test G.E.	Post-Test G.E.
OPTIMO	0.000%	100.000%
ACEPTABLE	18.182%	0.000%
MEDIO	81.818%	0.000%
BAJO	0.000%	0.000%

CRITICO	0.000%	0.000%

Fuente: Resultados del Pre-Test y Post-Test relativos a la seguridad

Fecha: 2015-07-03

De los resultados obtenidos de la comparación del pre-test y post-test que se han mostrado en la tabla 43 podemos concluir lo siguiente: Que tras aplicar el estímulo al grupo experimental se ha evidenciado una mejora significativa en el nivel de seguridad referente a la atención de los requerimientos de monitoreo de las cuentas de usuarios y sus privilegios. Esto se sustenta en que **el 100% de los sistemas y servicios consiguieron la categoría OPTIMO** en el post-test, mientras que en el pre-test los valores encontrados fueron del 18.182% (ACEPTABLE), un 81.818% (MEDIO), 0% (BAJO) respectivamente.

PRUEBA DE HIPOTESIS

Según Hernández Sampieri (2014), "las hipótesis se someten a prueba o escrutinio empírico para determinar si son apoyadas o refutadas, de acuerdo con lo que el investigador observa". (p. 117), y para ello se recurren a diferentes métodos estadísticos.

Para la presente investigación, se ha elegido la prueba de hipótesis t porque "la prueba t se utiliza para comparar los resultados de una preprueba con los resultados de una posprueba en un contexto experimental". (Hernández, 2014, p. 311).

Tabla 44
Prueba de Hipotesis t - Estadísticas de muestras emparejadas

		Media	N	Desviación estándar	Media de error estándar
Par 1	Pre-Test	3,54545	11	,934199	,281672
	Post-Test	1,00000	11	,000000	,000000

Fuente: SPSS

Tabla 45
Prueba de Hipotesis t - Correlaciones de muestras emparejadas

	N	Correlación	Sig.
Par 1 Pre-Test & Post-Test	11	.	.

Fuente: SPSS

Tabla 46

Prueba de Hipotesis t - Prueba de muestras emparejadas

	Diferencias emparejadas					t	gl	Sig. (bilateral)
	Media	Desviación estándar	Media de error estándar	95% de intervalo de confianza de la diferencia				
				Inferior	Superior			
Par 1 Pre-Test - Post-Test	2,545455	,934199	,281672	1,917851	3,173058	9,037	10	,000

Fuente: SPSS

La prueba muestra una media de 2,545455. Así mismo, con el 95% de intervalo de confianza se ha obtenido una significancia bilateral de 0.000. Puesto que 0.000 es menor a 0.05 procedemos a rechazar la hipótesis nula.

Por lo tanto, se argumenta que existe una mejora significativa en la revisión de los derechos de acceso a los usuarios en sistemas operativos Linux.

DISCUSION DE LOS RESULTADOS

La necesidad de resolver el problema de la gestión de usuarios mediante un mecanismo de intercambio de información rápido y seguro dentro de una red de computadoras, coincide en gran medida con el concepto de lo que representa el propósito de un protocolo descrito como “un acuerdo sobre el intercambio de paquetes y su significado entre programas que se comunican entre sí. Un protocolo es usualmente diseñado para resolver un problema específico” (Makofske, Donahoo, & Calvert, 2004, pág. 2). Los diferentes sistemas y servicios utilizados en los servidores de Petroperú en su sede Iquitos son accedidos mediante cuentas de usuarios, para lo cual es requisito poder identificarlos previamente antes de diseñar el protocolo SUMP que va a permitir gestionar esa información.

El uso de la dimensión metodológica de Selección propuesta por el autor de esta investigación, ha permitido delimitar el alcance que tiene el protocolo SUMP dentro de su entorno tecnológico básico y cumplir con el objetivo planteado. El enfoque de 3 pasos (Obtener población de servidores, Identificar sistemas y servicios y Elaborar muestra de trabajo) es un método simple pero efectivo que ha conseguido identificar a los diferentes sistemas y servicios que necesitan de un protocolo de gestión de usuarios.

Con la información de la población de servidores entregada por Petroperú, se comienza a tener una idea clara sobre la plataforma de desarrollo en la que debe enfocarse el protocolo, la cual resulta ser el sistema operativo Linux con un total de 11 servidores, lo que representa el 100% de la población.

Al identificar los sistemas y servicios desde la población de servidores, se ha conseguido categorizarlos por su necesidad de monitoreo mediante la aplicación del cuestionario referido en el Anexo 01 que está compuesto por 2 preguntas técnicas: ¿Usa Cuentas de Usuarios Locales? y ¿Usa Privilegios Locales?. Estas preguntas técnicas permitieron evaluar el campo denominado “Requiere Monitoreo y Control“, cuyo resultado fue que 16 sistemas y servicios no lo requerían y 15 sistemas y servicios si, entre los cuales se encontraron los siguientes: DB2(2), HTTP(1), Linux(8), MySQL(3), Samba(1).

Finalmente se elaboró una muestra de trabajo a partir de los 15 sistemas y servicios identificados, donde Petroperú les estableció una prioridad desde el cuestionario referido en el Anexo 02, y se ha podido restringir el ámbito de acción del protocolo a 2 subsistemas con prioridad alta: Linux y MySQL. Con esta reducción no se ve alterado el diseño del protocolo, simplemente se prioriza su atención hacia estos subsistemas que son los más críticos y utilizados en toda la sede Selva de la empresa Petroperú.

El uso de la dimensión metodológica de Evaluación para la revisión de los derechos de acceso de los usuarios propuesta por el autor de esta investigación, ha permitido un diagnóstico preciso mediante la valoración, a través de un Pre-Test, de 2 indicadores clave: la performance y la seguridad.

Para (Hernández, 2014) en el diseño experimental se aplica una preprueba o Pre-Test consiguiendo la ventaja de contar con un punto de referencia inicial en el que se evalúa el estado de la variable dependiente antes del estímulo.

El indicador de performance ha sido abordado con recomendaciones basadas en los controles 9.2.5 de la ISO/IEC 27002:2013, que pueden ser revisadas en el Anexo 03 y cuya medición realizada fue el tiempo en minutos que tomaba atender todos los requerimientos de revisión de cuentas de usuario y privilegios. De esta forma se obtuvo como resultado que el 72.727% de los sistemas y servicios tenían una frecuencia de atención entre 2 a 3 minutos, por lo que su evaluación categórica fue de ACEPTABLE. Por otro lado, el 27.273% restante tenían una frecuencia de atención entre 4 a 9 minutos, por lo que su evaluación categórica fue de MEDIO.

A su vez, el indicador de seguridad ha sido abordado con recomendaciones basadas en los controles 13.1.2 de la ISO/IEC 27002:2013, las cuales también se encuentran detalladas en el Anexo 03. Para este indicador la medición adoptada fue cualitativa, evaluando 3 aspectos relativos a la seguridad (autenticación, control de conexiones y encriptación) a través de su categorización. Los resultados indicaron que el 18.182% de los sistemas y servicios si utilizan mecanismos de autenticación, cuentan con un control de conexiones y utilizan una encriptación mayor o igual a 1024bits y menor o igual a 2048bits, por lo que

su evaluación categórica fue ACEPTABLE. Por otro lado, el 81.818% restante tenían sistemas y servicios que si utilizan mecanismos de autenticación, no cuentan con un control de conexiones y utilizan una encriptación mayor o igual a 1024bits y menor o igual a 2048bits, por lo que su evaluación categórica fue de MEDIO.

Estos resultados obtenidos a partir del Pre-Test permiten determinar el estado actual del proceso de revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en su sede Iquitos, con una categoría no menor a MEDIO ni mayor a ACEPTABLE.

El diseño del protocolo SUMP ha sido abordado desde la perspectiva del diseño de software “visto como la actividad del ciclo de vida del software donde se analizan los requerimientos para producir una descripción de la estructura interna del software que servirá de base para su construcción”. (IEEE Computer Society, 2014) y complementándolo con la metodología de diseño formal así como la inclusión de una metodología heurística que ha servido para la elaboración del documento RFC utilizado como input para la construcción de un prototipo del protocolo, ambas propuestas por (König, 2012).

Como resultado se puede apreciar la aplicación de las 3 fases del diseño de protocolos sugerida por (König, 2012) que son: la especificación del servicio, la especificación del protocolo y la especificación de la implementación. Cada una de ellas se discutirá a continuación.

Se ha cumplido con el diseño de una especificación del servicio para el nuevo protocolo, describiendo la interacción del servicio SUMP entre los usuarios, los proveedores y sus interfaces; utilizando técnicas de descripción formal como diagramas TS, diagramas de estado, el lenguaje de modelado Level S y el diagrama de especificación del servicio.

También se ha cumplido con el diseño de una especificación de protocolo describiendo las operaciones de cada entidad en respuesta a las órdenes de los usuarios así como de los mensajes de otras entidades mediante unidades de datos tal como lo define (Lai & Jirachiefpattana, 1998) y siguiendo el lenguaje de modelado Level P propuesto por (König, 2012) como técnica de descripción

formal. Con esto se consiguió describir la interacción de las entidades pares SUMP-CLI y SUMP-SRV, el orden de sus funciones y el formato de sus PDUs.

Se ha cumplido con el diseño de una especificación de implementación en forma de prototipo sugerido por (König, 2012) cubriendo sus 4 sub-fases (análisis de restricciones del sistema, implementación local relevante, elección del modelo de implementación y especificación de la implementación) con el objetivo de implementarlo en un entorno de ejecución concreto y validar los principales procedimientos del protocolo durante su diseño. Como resultado se tienen los detalles de las restricciones del sistema, los diagramas de arquitectura del servidor y del cliente SUMP, el mapeo de procesos y el documento RFC con los detalles técnicos necesarios para su construcción (ver Anexo 13).

Finalmente, el diseño del protocolo SUMP cumple con los lineamientos planteados por (Shannon & Weaver, 1963) al incorporar todos los elementos de un sistema general de comunicación. La **fuente de información** corresponde con los datos de usuarios y privilegios suministrada por los módulos Unix y MySQL, el **transmisor** corresponde con la entidad SUMPD-SRV encargada de cambiar el mensaje en señales y lo enviarlo al receptor a través del canal de comunicación, el **mensaje** en sí mismo corresponde a las estructuras Request y Response que utilizan las entidades, la **señal** que corresponde a la codificación del mensaje original con la intención de que pueda viajar por el canal de comunicación, la **señal recibida** que corresponde a la decodificación del mensaje alterado o no durante la transmisión, el **canal de comunicación** que corresponde al medio de transmisión utilizado en una red Ethernet y soportado por el protocolo TCP/IP así como los túneles SSH, la **fuente de ruido** que corresponde con la alteración no deseada de la señal ocurrida en el canal durante la transmisión y que está presente de forma inherente en toda comunicación TCP/IP como errores de checksum o alteraciones provocadas como secuestro de sesiones, el **receptor** que corresponde con la entidad SUMPD-CLI que reconstruye el mensaje enviado por el transmisor, y finalmente el **destino** que corresponde con el agente o usuario final que hace uso de los métodos de acceso del protocolo SUMP y quien es el que recibe el mensaje enviado.

Con todos estos resultados, y siguiendo la dimensión metodológica de Diseño se obtuvo una categoría de OPTIMO en el proceso de diseño del protocolo SUMP.

El diseño experimental según lo sostiene (Hernández, 2014), debe concluir con la aplicación de una prueba posterior al estímulo, que en este caso es el Post-Test. El resultado de esta postprueba, ha permitido compararlo con los resultados obtenidos en el Pre-Test y evaluar el desempeño de la variable dependiente.

Debido a que las mediciones tienen su origen en la dimensión metodológica de Evaluación, tal como se ha visto en los objetivos 2, esta comparación se divide también en los 2 indicadores clave: performance y seguridad.

Para el indicador de performance donde se utilizó el protocolo SUMP, se obtuvo que el 100.000% de sistemas y servicios que requerían la atención de sus requerimientos lo hicieran con tiempos de respuesta menores a 1 minuto. Estos resultados elevaron a la categoría de OPTIMO los tiempos de respuesta al grupo experimental.

Para el indicador de seguridad, se obtuvo que el 100% de los sistemas y servicios utilizan mecanismos de autenticación, contaron con un control de conexiones y utilizaron una encriptación mayor o igual a 4096bits, por lo que su evaluación categórica fue OPTIMO.

La comparación de los resultados obtenidos del pre-test y post-test es el paso final de la investigación que presenta el autor. Con estos resultados contrastados entre sí, se puede determinar el nivel de mejora que se produce en la revisión de los derechos de acceso de los usuarios en la empresa Petroperú en la sede IQUITOS.

Los resultados son contundentes para ambos indicadores. Para el indicador de performance se obtuvo la categoría OPTIMA en el 100% de los sistemas y servicios que aplicaron el post-test mientras que en el pre-test los valores encontrados fueron de 72.727% con categoría ACEPTABLE y 27.273% con categoría MEDIO. Y para el indicador de seguridad se obtuvo también la categoría OPTIMA en el 100% de los sistemas y servicios que aplicaron el post-

test versus el 18.182% con categoría ACEPTABLE, y 81.818% con categoría MEDIO que se obtuvieron en el pre-test.

Finalmente, con estos resultados podemos afirmar que la hipótesis planteada se ha logrado confirmar tal y como sigue: “Si se aplica el protocolo SUMP, entonces se mejora significativamente la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en la sede Iquitos”.

CAPITULO V
CONCLUSIONES Y SUGERENCIAS

5. CAPITULO V – CONCLUSIONES Y SUGERENCIAS

5.1. Conclusiones

1. Se identificaron los diferentes sistemas y servicios utilizados en Petroperú sede Iquitos que requieren de un protocolo de gestión de usuarios, mediante la aplicación de los cuestionarios de identificación y priorización de sistemas y servicios; obteniéndose como resultado que el 73.33% de los sistemas y servicios fueron considerados con un nivel de prioridad Alto, el 20.00% con un nivel de prioridad Medio y el 6.67% con un nivel de prioridad Bajo.

2. Se determinó, mediante la aplicación de un pre-test, el proceso actual de revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en su sede Iquitos; concluyendo que el 72.727% de los sistemas y servicios se encuentran en la categoría ACEPTABLE y un 27.273% en la categoría MEDIO en relación a su performance y por otro lado el 18.182% tiene un nivel ACEPTABLE de seguridad mientras que un 81.818% tiene un nivel MEDIO.

3. Se ha logrado diseñar el protocolo SUMP (Simple User Management Protocol) para mejorar la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux, teniendo en cuenta la metodología mixta propuesta por el autor.

4. Se evaluó el funcionamiento de la simulación protocolo SUMP para la revisión de los derechos de acceso de los usuarios en sistemas operativos Linux de la empresa Petroperú en su sede Iquitos; a través de la aplicación del post-test, obteniéndose como resultado que el 100% de los sistemas y servicios tienen la categoría de OPTIMO tanto en performance como en seguridad, consiguiendo así una mejora significativa.

5.2. Sugerencias

1. Realizar una revisión trimestral de todos los sistemas y servicios en todas las sedes de Petroperú que requieran de un protocolo de gestión de usuarios, debido a que los sistemas y servicios se encuentran en constante cambio. Con esta identificación trimestral se pueden proponer nuevos módulos para su inclusión en una nueva versión del protocolo SUMP.

2. Incorporar el soporte para plataformas Windows como parte del protocolo SUMP, para que la transmisión de datos de cuentas de usuarios y sus privilegios abarque un mayor dominio de acción.

3. Se recomienda a la gerencia de TI de Petroperú, que impulse el diseño y desarrollo de nuevos protocolos, que permitan la mejora continua de procesos automatizables de carácter técnico como es el caso del protocolo SUMP, para que contribuya en su eficiencia, calidad y seguridad en el tratamiento de su información.

4. Se recomienda a la jefatura de TI de Petroperú en la sede Iquitos, completar gradualmente la implementación del protocolo SUMP, para mejorar significativamente la revisión de los derechos de acceso de los usuarios en los sistemas operativos Linux de sus servidores en producción; como punto de partida para su adopción en toda la institución.

REFERENCIAS BIBLIOGRAFICAS

REFERENCIAS BIBLIOGRAFICAS

Barros Arteaga, S. (2013). *Diseño de Protocolos para redes inalambricas de sensores sobre LatinOS: Metodología y Ejemplos.*

Behar, D. (2008). *Metodología de la Investigación.* Shalom.

Bernal, C. (2010). *Metodología de la investigación* (Tercera ed.). Colombia: Pearson.

Blank, A. G. (2004). *TCP/IP Foundations.* San Francisco: SYBEX.

Bovet, D. P., & Cesati, M. (2006). *Understanding the Linux Kernel* (Third ed.). CA: O'Reilly.

Butcher, M. (2007). *Mastering OpenLDAP.* Birmingham: Packt Publishing.

Chandra, P., Messier, M., & Viega, J. (2002). *Network Security with OpenSSL* (First ed.). CA: O'Reilly.

Corbet, J., Rubini, A., & Kroah, G. (2005). *Linux Device Drivers* (Third ed.). CA: O'Reilly.

Davies, J. (2011). *Implementing SSL/TLS Using Cryptography and PKI.* Indiana: Wiley Publishing, Inc.

Flanagan, H., & Ginoza, S. (s.f.). *RFC 7322 - RFC Style Guide.* Recuperado el 18 de Agosto de 2017, de RFC Editor: http://www.rfc-editor.org/pdfrfc/rfc7322.txt.pdf

Fuentelsaz, C., Pulpón, A., & Icard, M. (2006). *Elaboración y Presentación de un Proyecto de Investigación y una Tesina.* Barcelona: Publicacions i Edicions Universitat de Barcelona.

Fusco, J. (2007). *The Linux Programmer's Toolbox.* Massachusetts: Prentice Hall.

Halsall, F. (2005). *Computer Networking and the Internet* (Fifth ed.). Harlow: Pearson.

Hanson, D. R. (1997). *C Interfaces and Implementations Techniques for Creating reusable Software.* Massachusetts: Addison Wesley Longman Inc.

Hernández, R. (2014). *Metodología de la Investigación* (Sexta ed.). Mexico: McGraw-Hill.

Hurtado de Barrera, J. (2000). *Metodología de la Investigación Holística* (Tercera ed.). Caracas: Fundación Sypal.

IBM. (2013). *z/OS MVS Using the Subsystem Interface* (Version 2 Release 1 ed.). IBM Corporation.

IEEE Computer Society. (2014). *SWEBOK® v3.0 Guide to the Software Engineering Body of Knowledge.*

IETF,IRTF,IAB. (s.f.). *RFC Editor*. Recuperado el 18 de Agosto de 2017, de https://www.rfc-editor.org

ISO/IEC. (2013). *Information technology — Security techniques — Code of practice for information security controls* (Second ed.).

Jang, M. (2011). *RHCSA/RHCE Red Hat Linux Certification Study Guide* (Sixth ed.). McGraw-Hill.

Joyanes, L., & Zahonero, I. (2001). *Programación en C, Metodología, algoritmos y estructura de datos.* Madrid: McGraw Hill.

Kalle Dalheimer, M., & Welsh, M. (2006). *Running Linux* (Fifth ed.). CA: O'Reilly Media.

Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language* (Second ed.). Massachusetts: Prentice Hall.

Kerrisk, M. (2010). *The Linux programming interface : a Linux and UNIX system programming handbook.* San Francisco: No starch Press.

King, K. N. (2008). *C Programming a Modern Approach* (Second ed.). Georgia: Norton.

Klemens, B. (2014). *21st Century C* (Second ed.). CA: O'Reilly.

Kochan, S. G. (2005). *Programming in C* (Third ed.). Indiana: Sam's Publishing.

König, H. (2012). *Protocol Engineering.* Berlin: Springer.

Lai, R., & Jirachiefpattana, A. (1998). *Communication protocol specification and verification.* Springer Science+Business Media,LLC.

Liu, Y., Yue, Y., & Guo, L. (2011). *UNIX Operating System - The Development Tutorial via UNIX Kernel Services.* Beijin: Springer.

Loshin, P. (1999). *Essential Email Standards: RFCs and Protocols Made Practical.* John Wiley & Sons.

Love, R. (2010). *Linux Kernel Development* (Third ed.). Indiana: Pearson Education.

Love, R. (2013). *Linux System Programming* (Second ed.). CA: O'Reilly.

Lucas, M. W. (2012). *SSH Mastery: OpenSSH, PuTTY, Tunnels and Keys.* Tilted Windmill Press.

Makofske, D. B., Donahoo, M. J., & Calvert, K. L. (2004). *TCP/IP Sockets in C# Practical Guide for Programmers* (First ed.). San Francisco: Morgan Kaufmann.

Masters, J., & Blum, R. (2007). *Professional Linux Programming.* Indiana: Wiley Publishing, Inc.

Matloff, N., & Salzman, P. J. (2008). *The art of debugging with GDB, DDD and Eclipse.* San Francisco: No Starch Press.

Matthew, N., & Stones, R. (2008). *Beginning Linux Programming* (Fourth ed.). Indiana: Wiley Publishing, Inc.

Mauro, D. R., & Schmidt, K. J. (2005). *Essential SNMP* (Second ed.). CA: O'Reilly.

McMillan, T. (2012). *Cisco Networking Essentials* (First ed.). Indiana: Sybex.

Mitchell, M., Oldham, J., & Samuel, A. (2001). *Advanced Linux Programming.* Indiana: New Riders Publishing.

MySQL-AB. (2006). *MySQL® Administrator's Guide and Language Reference* (First ed.). Seatle: MySQL Press.

Naghi Namakforoosh, M. (2014). *Metodología de la Investigación.* Mexico: Limusa.

Peterson, L. L., & Davie, B. S. (2012). *Computer Networks a systems approach* (Fifth ed.). Burlington: Morgan Kaufmann.

Rodríguez, E. A. (2005). *Metodología de la Investigación.* Mexico: Universidad Juarez Autonoma de Tabasco.

Rose, M. (Noviembre de 2001). *RFC 3117 - On the Design of Application Protocols.* Recuperado el 18 de Agosto de 2017, de RFC Editor: https://tools.ietf.org/pdf/rfc3117

Schwartz, B., Zaitsev, P., Tkachenko, V., Zawodny, J. D., Lentz, A., & Balling, D. J. (2008). *High Performance MySQL* (Second ed.). CA: O'Reilly Media.

Shannon, C. E., & Weaver, W. (1963). *The Mathematical Theory of Communication.* University of Illinois.

Sharp, R. (2008). *Principles of Protocol Design.* Denmark: Springer.

Shasankar, K. (2013). *Zend Framework 2.0 by Example Beginner's Guide* (First ed.). Birmingham: Packt Publishing.

Stevanovic, M. (2014). *Advanced C and C++ Compiling.* San Francisco: Apress.

Tanenbaum, A. S., & Wetherall, D. J. (2011). *Redes de Computadoras* (Cuarta ed.). Amsterdam: Pearson Prentice Hall.

UCLA. (18 de Setiembre de 2014). *Leonard Kleinrock's Home Page*. Recuperado el 12 de Agosto de 2015, de https://www.lk.cs.ucla.edu/index.html

Valade, J. (2005). *Spring into Linux* (First ed.). New York: Pearson Education.

Viega, J., & Messier, M. (2003). *Secure Programming Cookbook for C and C++.* CA: O'Reilly.

W3Techs. (18 de Mayo de 2017). *Usage Statistics and Market Share of Unix for Websites, May 2017*. Recuperado el 18 de Mayo de 2017, de Usage Statistics and Market Share of Unix for Websites, May 2017: https://w3techs.com/technologies/overview/operating_system/all

Wikipedia. (25 de Enero de 2015). *User space - Wikipedia*. Recuperado el 2015 de Enero de 2015, de User space - Wikipedia: https://en.wikipedia.org/w/index.php?title=User_space&diff=644105388&oldid=644105192

Wikipedia. (14 de Mayo de 2017). *Modelo OSI - Wikipedia, la enciclopedia libre*. Recuperado el 18 de Mayo de 2017, de Modelo OSI - Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Modelo_OSI

Yaghmour, K., Masters, J., Ben-Yossef, G., & Gerum, P. (2008). *Building Embedded Linux Systems* (Second ed.). CA: O'Reilly.

ANEXOS

Anexo 01 - GUIA TECNICA Y CUESTIONARIO DE IDENTIFICACIÓN DE SISTEMAS Y SERVICIOS

GUIA TECNICA

Descripción

El presente documento es una guía técnica para la aplicación del cuestionario de identificación de sistemas y servicios que se vienen utilizando en Petroperú-Iquitos.

Cuestionario de Identificación de Sistemas y Servicios

Servidor	**Sistema / Servicio**	**Usa Cuentas de Usuarios Locales**	**Usa Privilegios**	**Requiere Monitoreo y Control**
Caracteres	Caracteres	SI/NO	SI/NO	SI/NO

Dónde:

Servidor: Es el nombre del servidor Linux que se utiliza dentro de la infraestructura tecnológica de Petroperú-Iquitos

Sistema / Servicio: Es el nombre del servicio, sistema o subsistema que se encuentra desplegado en el servidor asociado

Usa Cuentas de Usuario Locales: Se debe evaluar si el sistema o servicio está configurado de tal modo que requiere de un mecanismo de autenticación de cuentas de usuario que residen en el mismo servidor para que pueda brindar la información solicitada por el empleado.

Usa Privilegios: Se debe evaluar si el sistema o servicio funciona con esquemas de privilegios según el grado o perfil de usuario, de tal forma que cierta información quede restringida para cierto grupo de usuarios

Requiere Monitoreo y Control: Se debe evaluar con el personal del área de TI de Petroperú si el servicio asociado, que soporta cuenta de usuarios locales y privilegios, requiere la inclusión en el sistema de monitoreo que simule el funcionamiento de protocolo propuesto en este proyecto de investigación.

ESCUELA DE POSTGRADO
UNIVERSIDAD CÉSAR VALLEJO

CUESTIONARIO DE IDENTIFICACIÓN DE SISTEMAS Y SERVICIOS

Lugar: ……..

Fecha: …… **Hora**: ……

Descripción:

El presente cuestionario está dirigido al personal encargado del área de Tecnología y Comunicaciones con el objetivo de identificar todos los sistemas y servicios que se viene utilizando en los servidores de Petroperú en su sede Iquitos.

La información recopilada en este cuestionario será compilada en el documento "Requerimiento de Información sobre Sistemas y Servicios" que será utilizada de manera confidencial y sólo con propósitos académicos por parte del encuestador.

Encuestador: Fernando Díaz Sánchez

Instrucciones:

Contestar las siguientes preguntas con el mayor rigor y veracidad posible. Tener en cuenta que solo deben ser considerados aquellos servidores cuyo sistema operativo sea Red Hat Enterprise Linux.

1. ¿Cuál es el nombre de los servidores Linux dentro de la infraestructura tecnológica de Petroperú-Iquitos y que sistemas o servicios brindan?

Servidor	Sistemas o Servicios

2. Por cada sistema o servicio indicado en la pregunta 1. ¿El sistema o servicio usa cuentas de usuarios locales?

Servidor	Usa Cuentas de Usuarios Locales
	SI( ) NO( )

	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )

3. Por cada sistema o servicio indicado en la pregunta 1. ¿El sistema o servicio usa privilegios de tal forma que cierta información quede restringida para algún grupo de usuarios?

Servidor	**Usa Privilegios**	
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )

4. Por cada sistema o servicio indicado en la pregunta 1. ¿El sistema o servicio es crítico, requiriendo un monitoreo y control con mayor prioridad?

Servidor	**Requiere Monitoreo y Control**	
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )
	SI( )	NO( )

Anexo 02 – GUIA TECNICA Y CUESTIONARIO DE PRIORIZACION DE SISTEMAS Y SERVICIOS

GUIA TECNICA

Descripción

El presente documento es una guía técnica para la aplicación del cuestionario de priorización de sistemas y servicios que se vienen utilizando en Petroperú-Iquitos.

Cuestionario de Identificación de Prioridad

Servidor	**Sistema o Servicio**	**Descripción**	**Prioridad**
Caracteres	Caracteres	Caracteres	Alto|Medio|Bajo

Dónde:

Servidor: Es el nombre del servidor que se utiliza dentro de la infraestructura tecnológica de Petroperú-Iquitos

Sistema / Servicio: Es el nombre del servicio, sistema o subsistema que se encuentra desplegado en el servidor asociado

Descripción: Es la descripción del propósito del sistema, subsistema o servicio utilizado.

Prioridad: Es la evaluación de la prioridad, que se debe tener en cuenta al momento de considerar la implementación del protocolo propuesto. Los valores de prioridad son Alto, Medio y Bajo.

ESCUELA DE POSTGRADO
UNIVERSIDAD CÉSAR VALLEJO

CUESTIONARIO DE PRIORIZACION DE SISTEMAS Y SERVICIOS

Lugar: ……..

Fecha: …… **Hora**: ……

Descripción:

El presente cuestionario está dirigido al personal encargado del área de Tecnología y Comunicaciones con el objetivo de medir el nivel de prioridad de todos los sistemas y servicios que se viene utilizando en los servidores de Petroperú en su sede Iquitos, que se identificaron en el Cuestionario de Identificación de Sistemas y Servicios.

La información recopilada en este cuestionario será compilada en el documento "Requerimiento de Información sobre Priorización de Sistemas y Servicios" que será utilizada de manera confidencial y sólo con propósitos académicos por parte del encuestador.

Encuestador: Fernando Díaz Sánchez

Instrucciones:

Contestar las siguientes preguntas con el mayor rigor y veracidad posible.

1. ¿Cuál es el propósito de los siguientes sistemas o servicios que se utilizan dentro de la infraestructura tecnológica de Petroperú-Iquitos?

Servidor	Sistema o Servicio	Descripción
Proxy Server	Linux	
Mail Server	Linux	
SMB Server	Linux	
SMB Server	Samba	
Central Server	Linux	
Central Server	MySQL	
Central Server	HTTP	
Backup Server	Linux	
Backup Server	DB2	
Syslog Server	Linux	
Monitor Server 1	Linux	
Monitor Server 1	MySQL	
Monitor Server 2	MySQL	
Directory Server	Linux	
Directory Server	DB2	

2. ¿Cuál es el nivel de prioridad de los siguientes sistemas o servicios que se utilizan dentro de la infraestructura tecnológica de Petroperú-Iquitos? Marque con una X en el nivel que corresponda.

Servidor	**Sistema o Servicio**	**NIVEL DE PRIORIDAD**		
		ALTO	**MEDIO**	**BAJO**
Proxy Server	Linux			
Mail Server	Linux			
SMB Server	Linux			
SMB Server	Samba			
Central Server	Linux			
Central Server	MySQL			
Central Server	HTTP			
Backup Server	Linux			
Backup Server	DB2			
Syslog Server	Linux			
Monitor Server 1	Linux			
Monitor Server 1	MySQL			
Monitor Server 2	MySQL			
Directory Server	Linux			

Anexo 03 - PRE-TEST

ATENCION DE REQUERIMIENTOS DE CUENTAS DE USUARIO

Descripción

El presente pre-test está dirigido al encargado de administrar la red de servidores en Petroperú IQUITOS para determinar el estado actual del proceso de revisión de cuentas de usuario y privilegios basado en el control 9.2.5 y 13.1.2 de la ISO/IEC 27002:2013.

Las respuestas deben darse en base a la experiencia práctica, y deben ser validadas en un entorno de producción durante requerimientos oficiales del personal de Petroperú-Iquitos. Muchas gracias por su cooperación.

Servidor: ……

Sistema o Servicio: ……..

Instrucciones:

Lea detenidamente y responda las siguientes preguntas

A. Relativo a la Performance:

¿Cuánto tiempo le toma (en segundos) realizar la siguiente actividad durante un requerimiento de revisión de cuentas de usuario y privilegios?

1. Login al sistema operativo: ……..
2. Observar información del usuario: ……
3. Observar información de privilegios: ……
4. Observar información sobre el grupo del usuario: ……..
5. Crear un nuevo usuario o grupo: ……
6. Modificar un usuario o grupo: ……..
7. Eliminar un usuario o grupo: ……
8. Validar la información observada: ……...

Total: ……

B. Relativo a la Seguridad

1. Autenticación: ¿Utiliza mecanismos de autenticación para acceder al servidor que brinda el sistema o servicio?
   SI ( ) NO ( )

2. Control de Conexiones: ¿Utiliza mecanismos de control de conexiones de red para que solo acceda personal autorizado al servidor que brinda el sistema o servicio (p.e. firewall rules, acl, etc)?

   SI ( ) NO ( )

3. Encriptación: ¿Qué nivel de encriptación utiliza para la transmisión de información por la red en el servidor que brinda el sistema o servicio?

   Ninguno ( )

   <=512bits ( )

   <=1024bits ( )

   <=2048bits ( )

   >=4096bits ( )

Anexo 04 - POST-TEST

ATENCION DE REQUERIMIENTOS DE CUENTAS DE USUARIO CON EL PROTOCOLO SUMP

Descripción

El presente post-test está dirigido al encargado de administrar la red de servidores en Petroperú IQUITOS para determinar el comportamiento del protocolo SUMP para el proceso de revisión de cuentas de usuario y privilegios basado en el control 9.2.5 y 13.1.2 de la ISO/IEC 27002:2013.

Las respuestas deben darse en base a una simulación de los servidores en un entorno virtualizado entregado por el investigador, y no requieren ser validadas en un entorno de producción durante requerimientos oficiales del personal de Petroperú-Iquitos. Muchas gracias por su cooperación.

Servidor: ……

Sistema o Servicio: ……..

Instrucciones:

Lea detenidamente y responda las siguientes preguntas

A. **Relativo a la Performance:**
   ¿Cuánto tiempo le tomó (en segundos) realizar la siguiente actividad durante un requerimiento de revisión de cuentas de usuario y privilegios ficticio con el uso del protocolo SUMP?

   1. Login al sistema operativo: ……..
   2. Observar información del usuario: …….
   3. Observar información de privilegios: ……..
   4. Observar información sobre el grupo del usuario: ……...
   5. Crear un nuevo usuario o grupo: ……
   6. Modificar un usuario o grupo: ……..
   7. Eliminar un usuario o grupo: ……
   8. Validar la información observada: ……

   Total: ……

B. **Relativo a la Seguridad**

1. Autenticación: ¿Utiliza mecanismos de autenticación para acceder al servidor que brinda el sistema o servicio?
   SI ( ) NO ( )

2. Control de Conexiones: ¿Utiliza mecanismos de control de conexiones de red para que solo acceda personal autorizado al servidor que brinda el sistema o servicio (p.e. firewall rules, acl, etc)?

   SI ( ) NO ( )

3. Encriptación: ¿Qué nivel de encriptación utiliza para la transmisión de información por la red en el servidor que brinda el sistema o servicio?

   Ninguno ( )

   <=512bits ( )

   <=1024bits ( )

   <=2048bits ( )

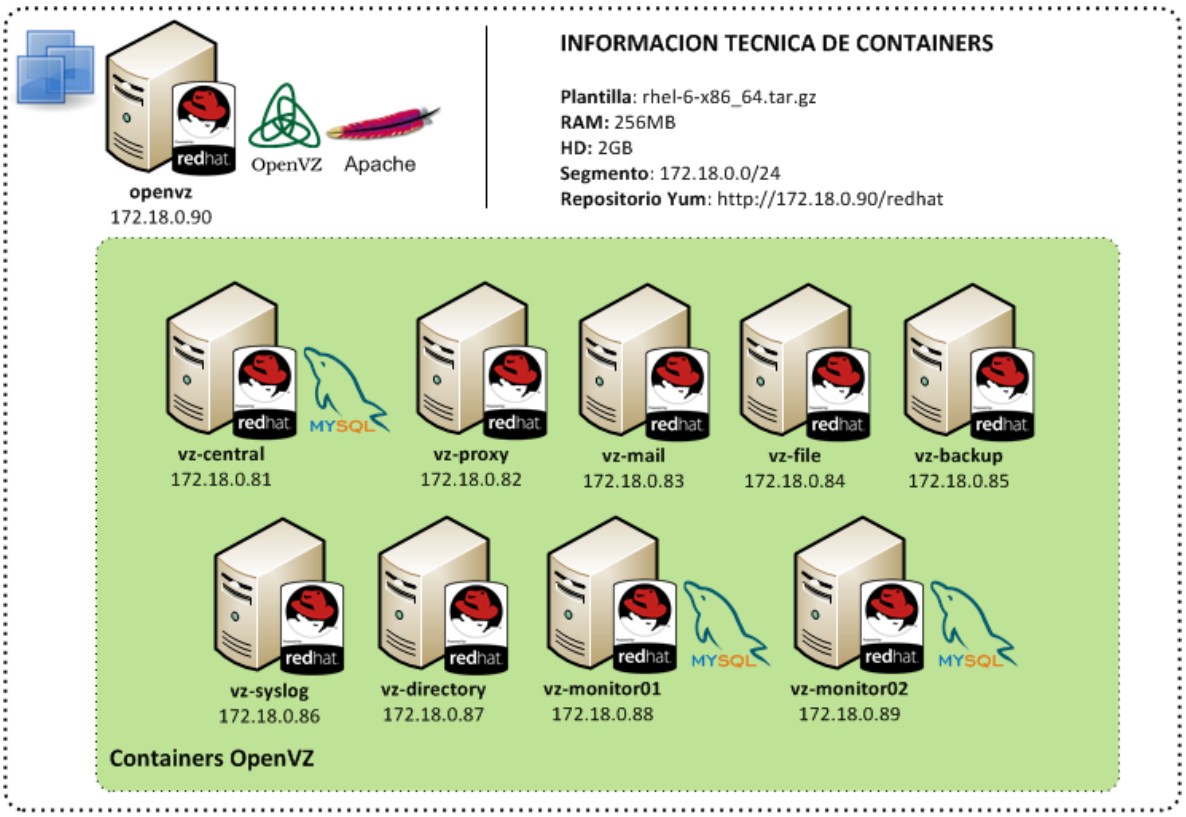
   >=4096bits ( )

Anexo 05 – INFRAESTRUCTURA VIRTUAL

Descripción

Este documento contiene el diagrama de la infraestructura virtual que será utilizada para la simulación del funcionamiento del protocolo sump, la cual será implementada en los anexos 06, 07, y 08.

INFRAESTRUCTURA VIRTUAL UTILIZADA



Anexo 06 - INSTALACION DE OPENVZ

Descripción

Este documento contiene las instrucciones para instalar y configurar el servidor OpenVZ donde se alojarán los servidores destinados para la simulación.

Requerimientos Mínimos

- ✓ Software VMware ESX 5 previamente instalado
- ✓ Imagen del sistema operativo Red Hat Enterprise Linux 6
- ✓ Kernel Updates
- ✓ Acceso a internet

A - INSTALAR EL SISTEMA OPERATIVO RED HAT ENTERPRISE LINUX 6

Paso 01 – Encender la máquina virtual

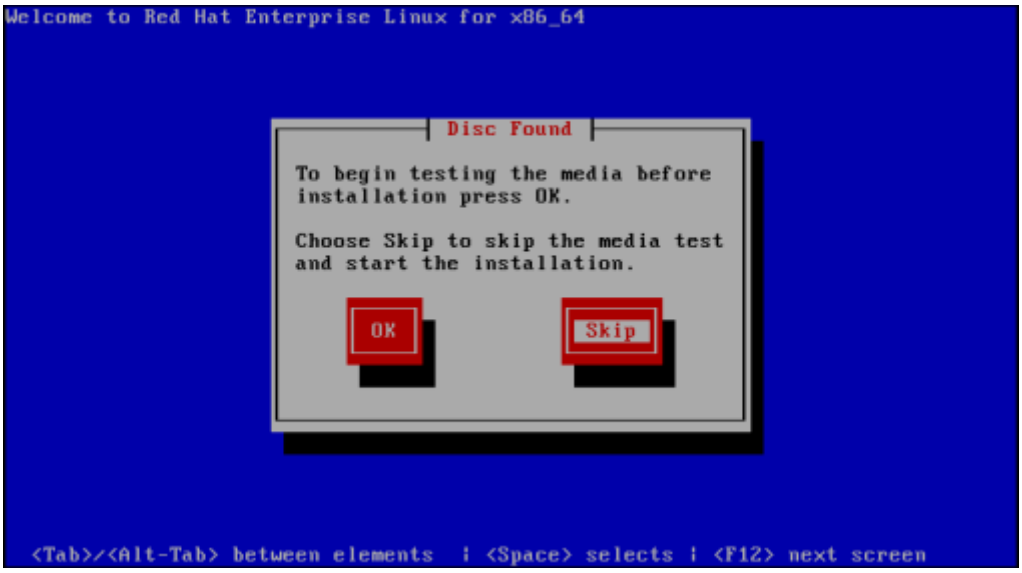
La máquina virtual debe ser encendida desde la consola de VMware ESX server.

Paso 02 – Elegir el tipo de instalación

Seleccionar “Install or upgrade an existing system” y presionar Enter.

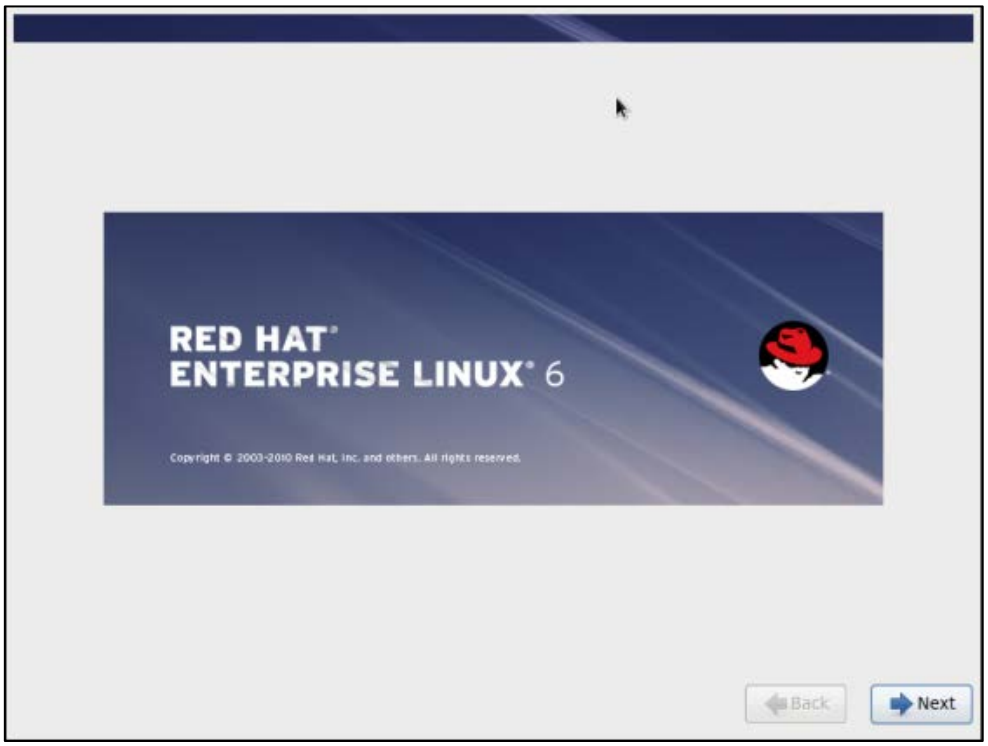
Paso 03 – Evitar el test del medio de instalación

Seleccionar el menú Skip para evitar el test del medio de instalación

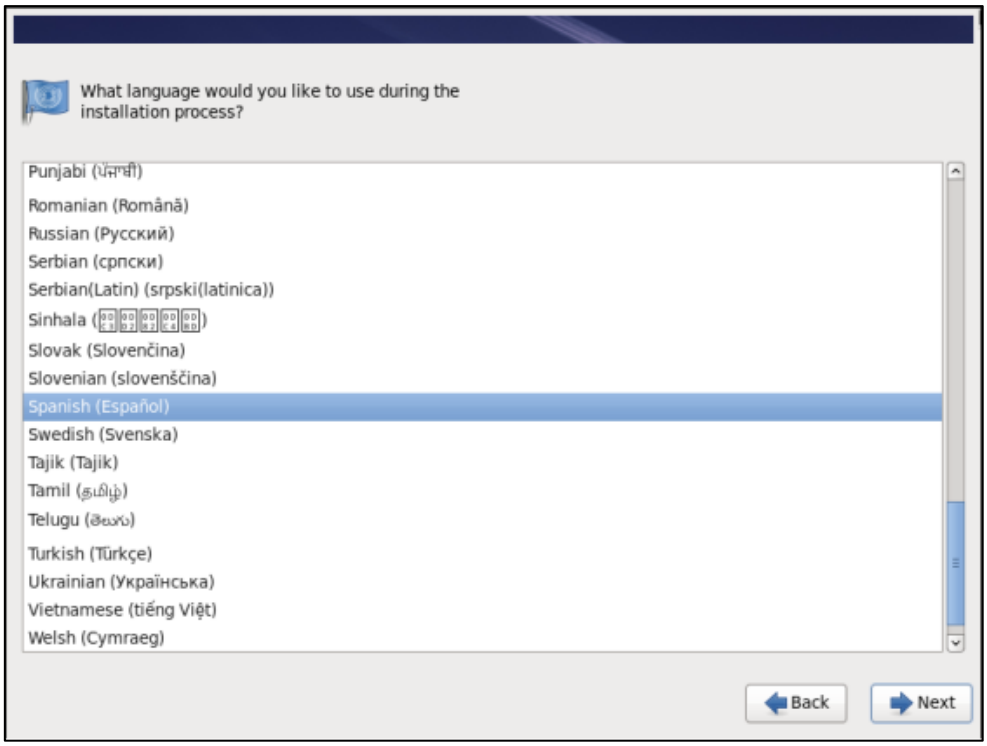


Paso 04 – Iniciar el proceso de instalación

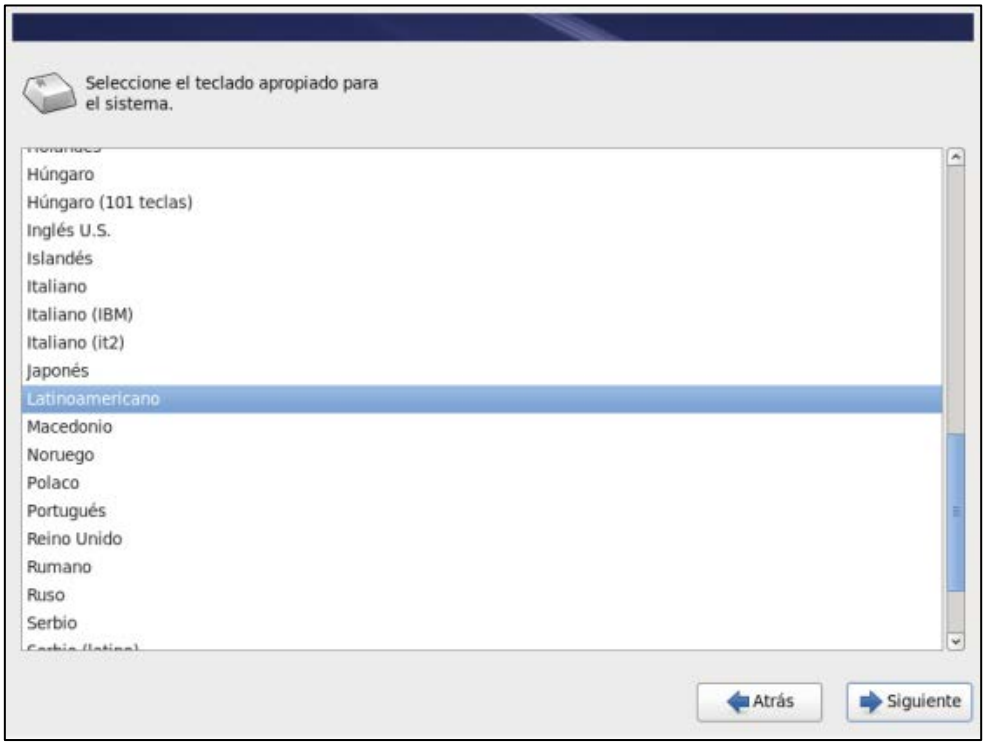
En la pantalla de bienvenida, dar click en Next



Paso 05 – Seleccionar el idioma Español durante el proceso de instalación

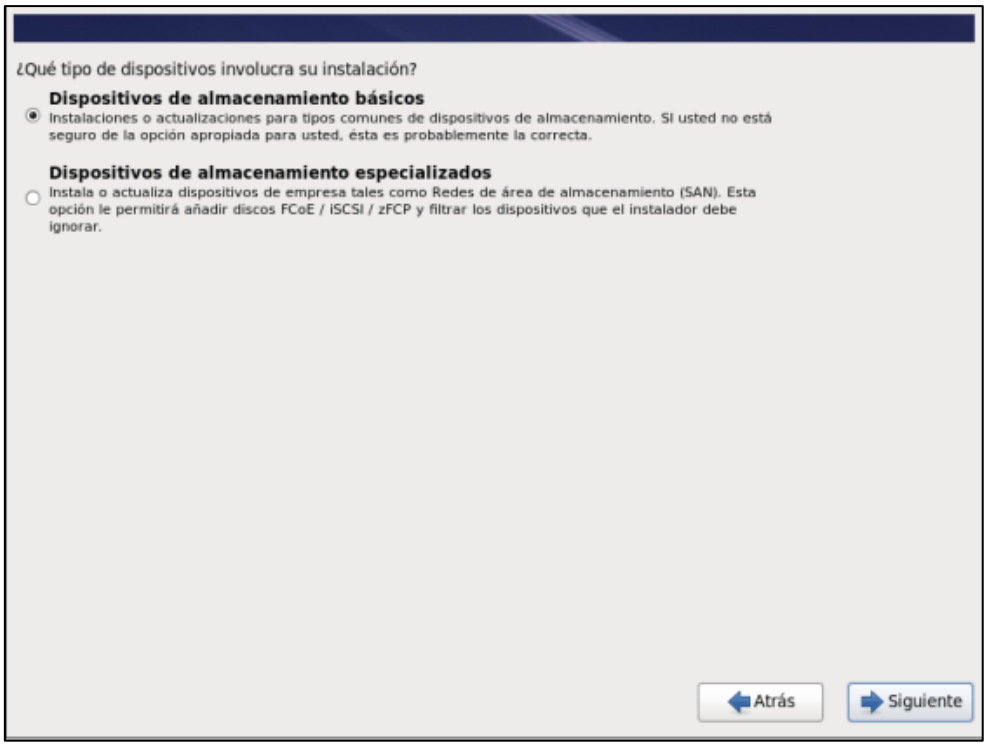


Paso 06 – Seleccionar el teclado Latinoamericano para el servidor



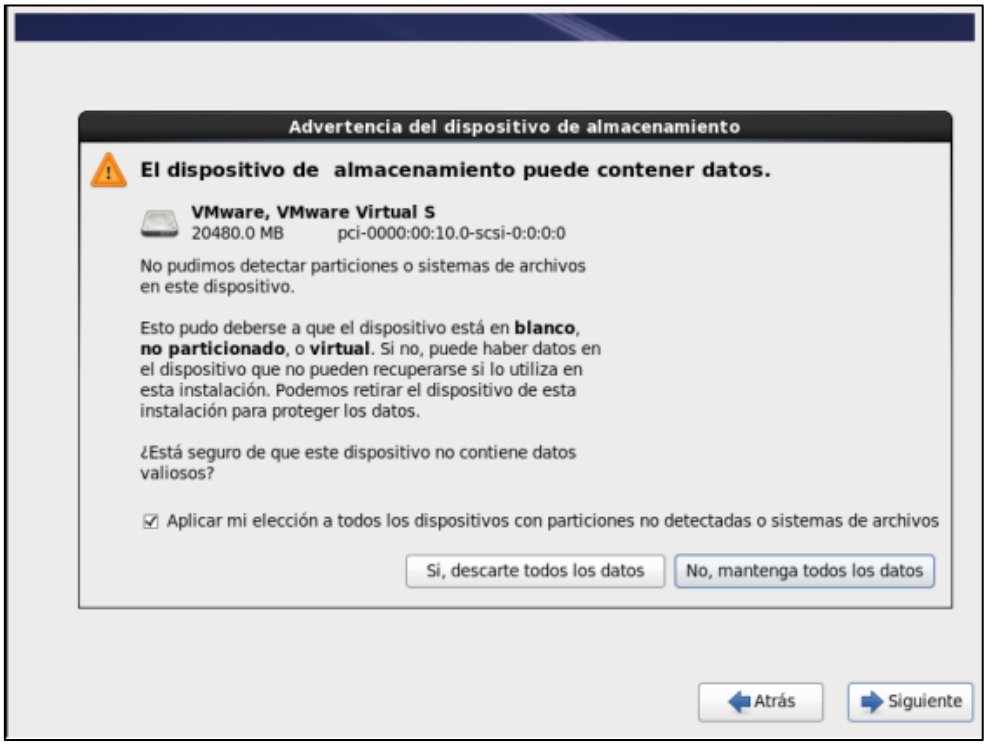
Paso 07 – Seleccionar el tipo de dispositivos de almacenamiento

Elegir “Dispositivos de almacenamiento básicos”

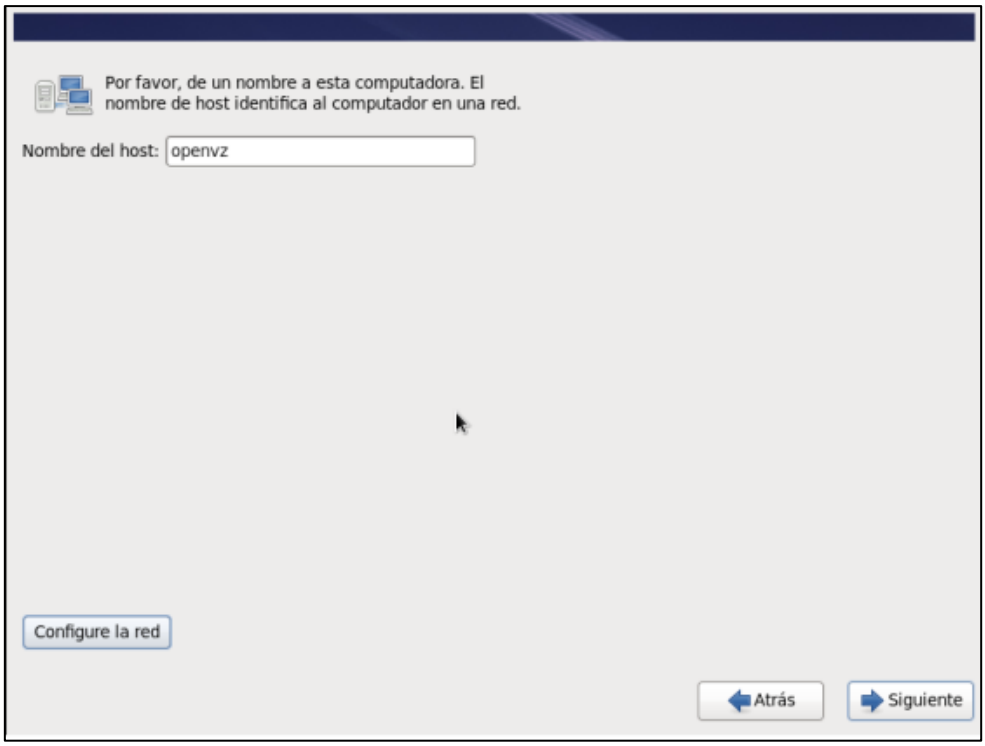


Paso 08 – Eliminar los datos del medio de almacenamiento

Si no se detectan particiones en el disco elegir “Si, descarte todos los datos”

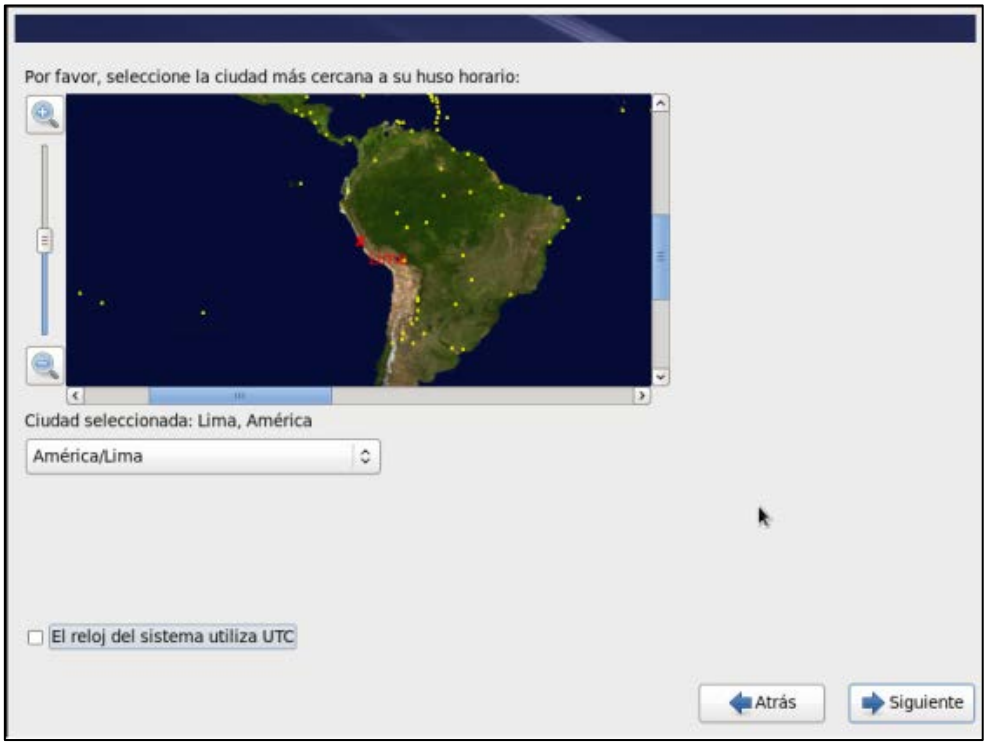


Paso 09 – Configurar el nombre del host como openvz

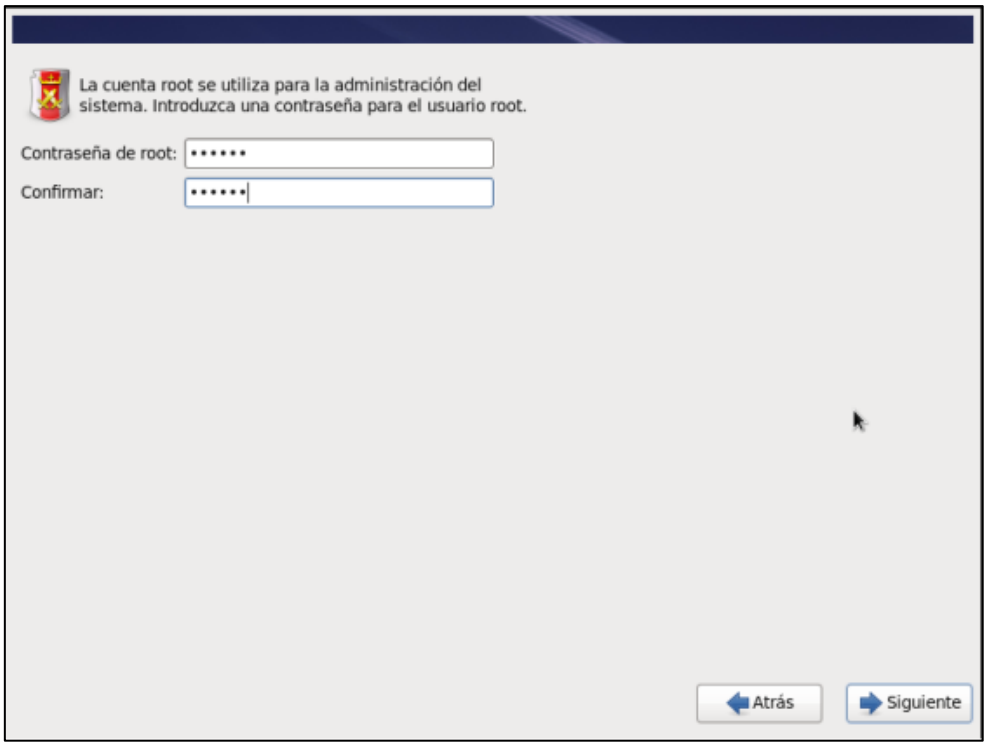


Paso 10 – Configurar la zona horaria

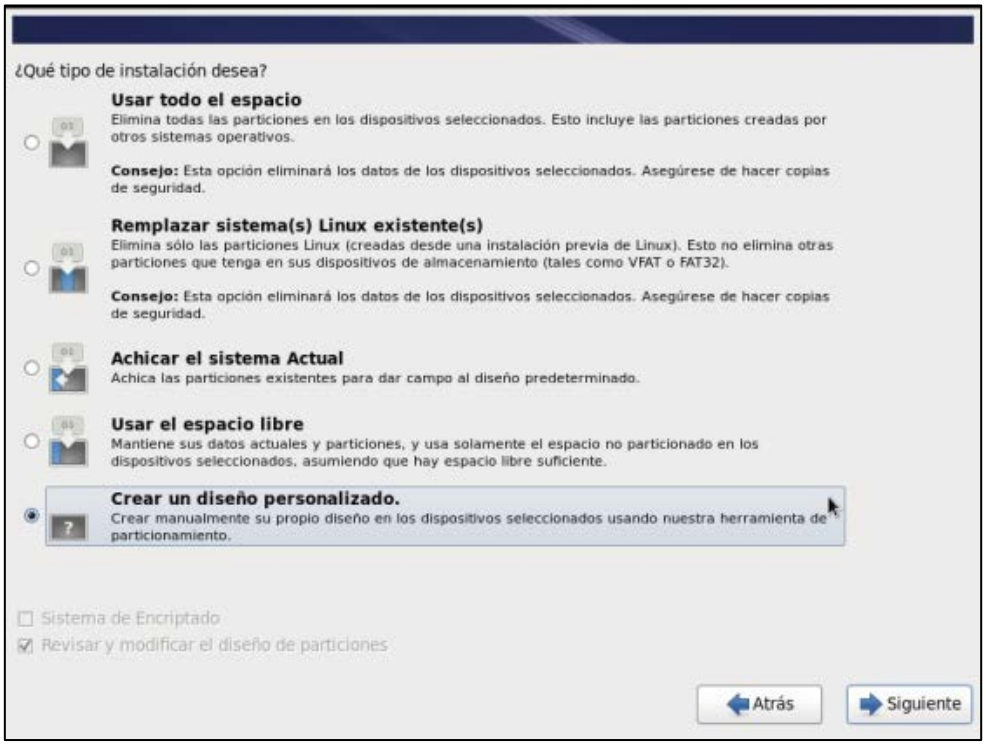
Seleccionar “América/Lima” y desactivar la opción “el reloj del sistema utiliza UTC”



Paso 11 – Establecer la clave del usuario root

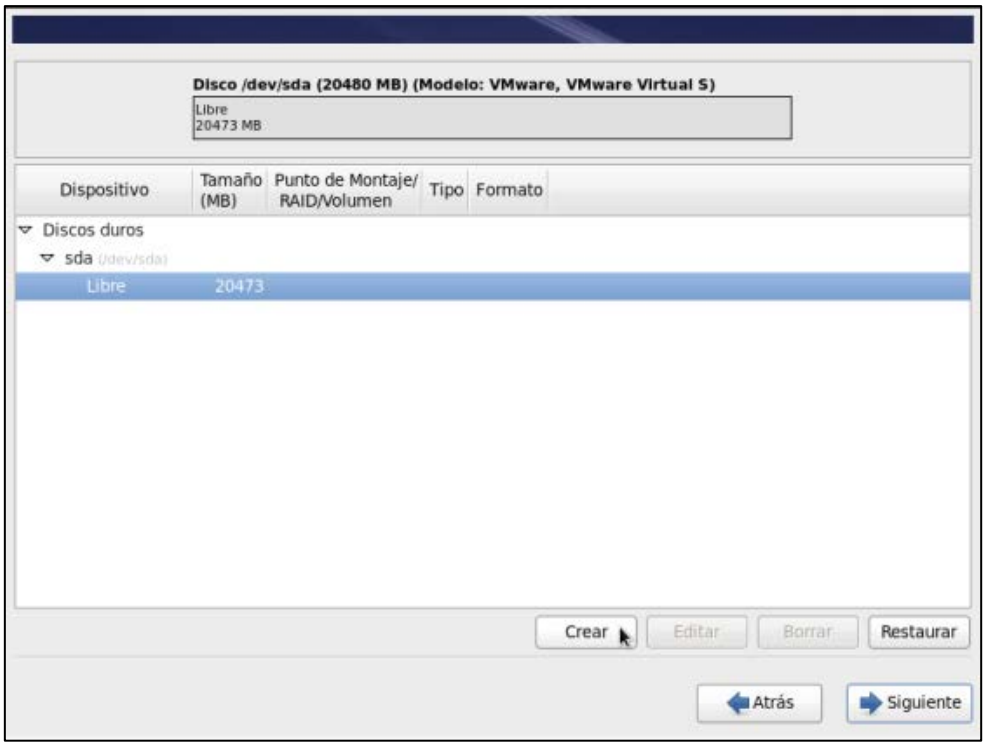


Paso 12 – Seleccionar el tipo de particionamiento personalizado



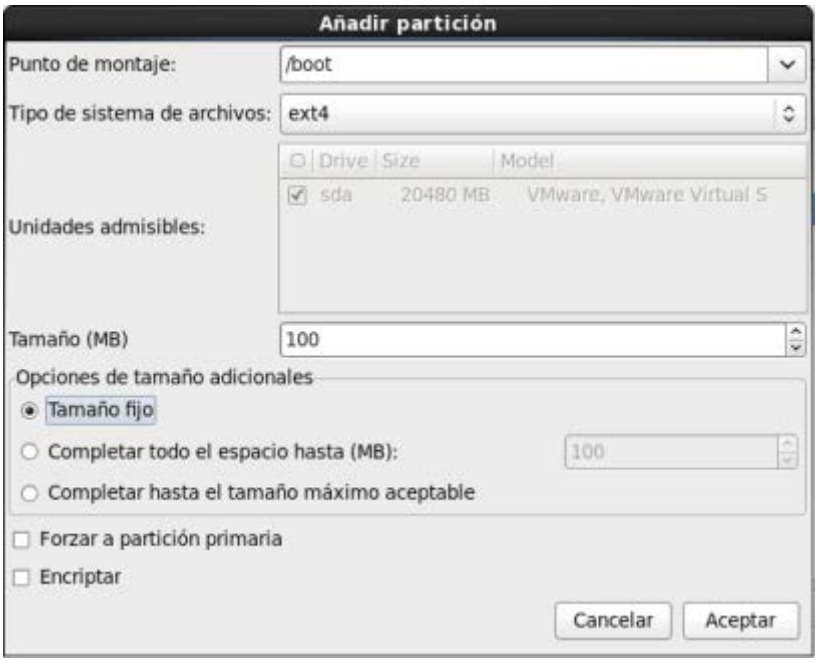
Paso 13 – Crear la partición /boot

Seleccionar el espacio libre y presionar el botón “Crear”



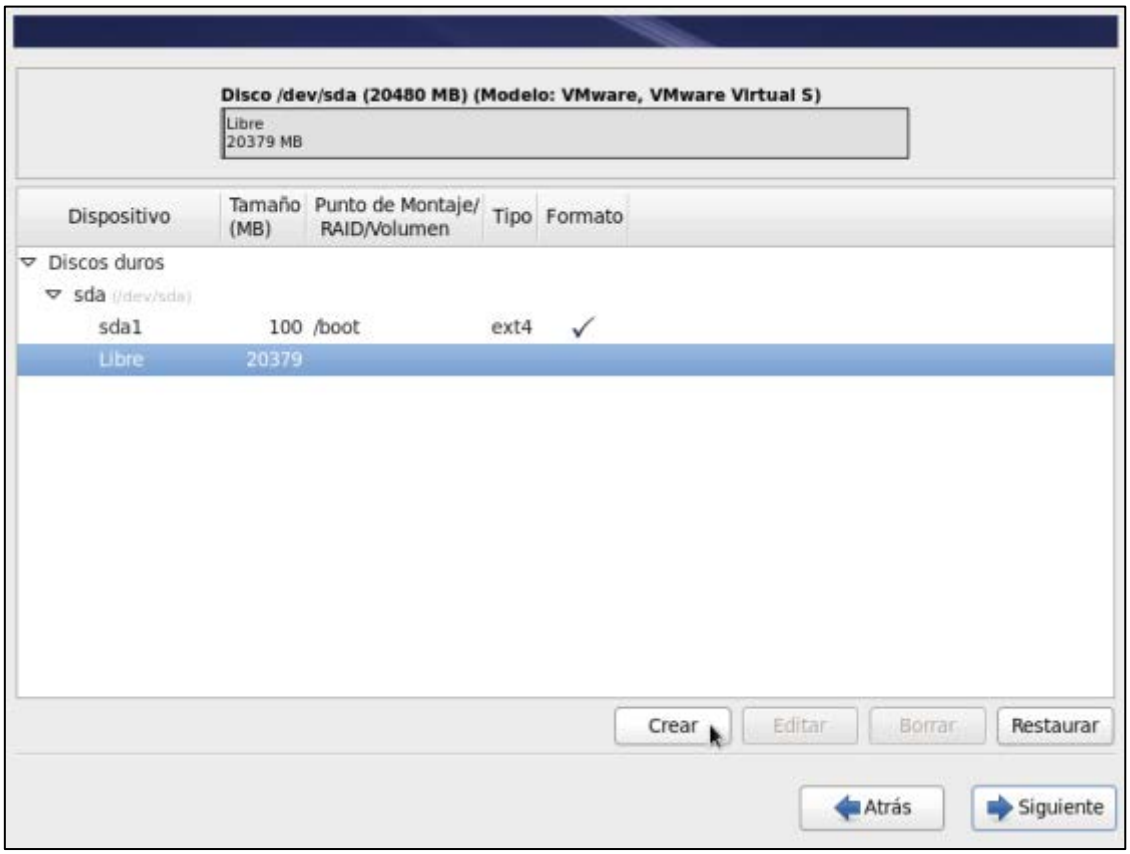
Luego seleccionar Partición estándar y dar click en “Crear”

En el cuadro Añadir partición colocamos el punto de montaje /boot, el tipo de sistema de archivos ext4 con 100MB de tamaño fijo según la imagen



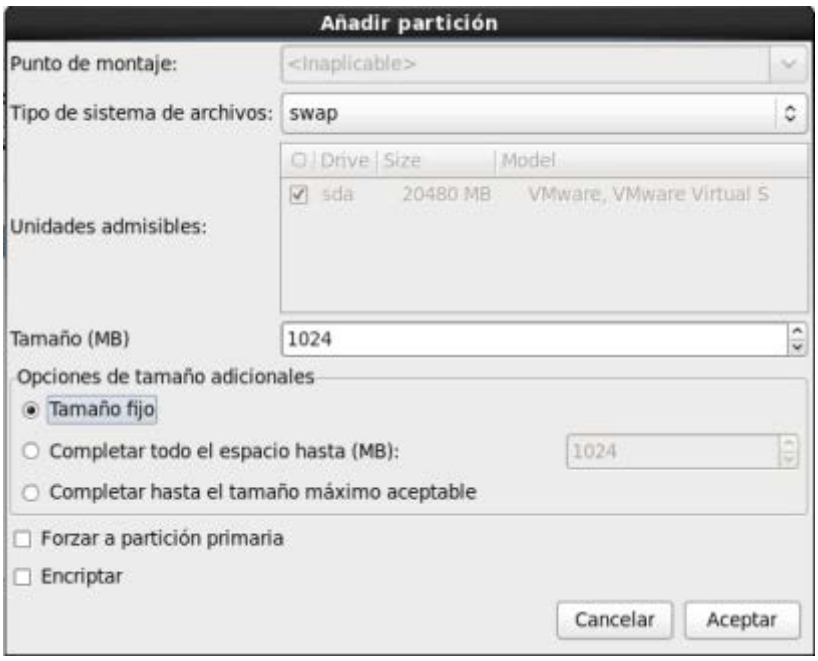
Paso 14 – Crear la partición swap

Seleccionar el espacio libre y presionar el botón “Crear”



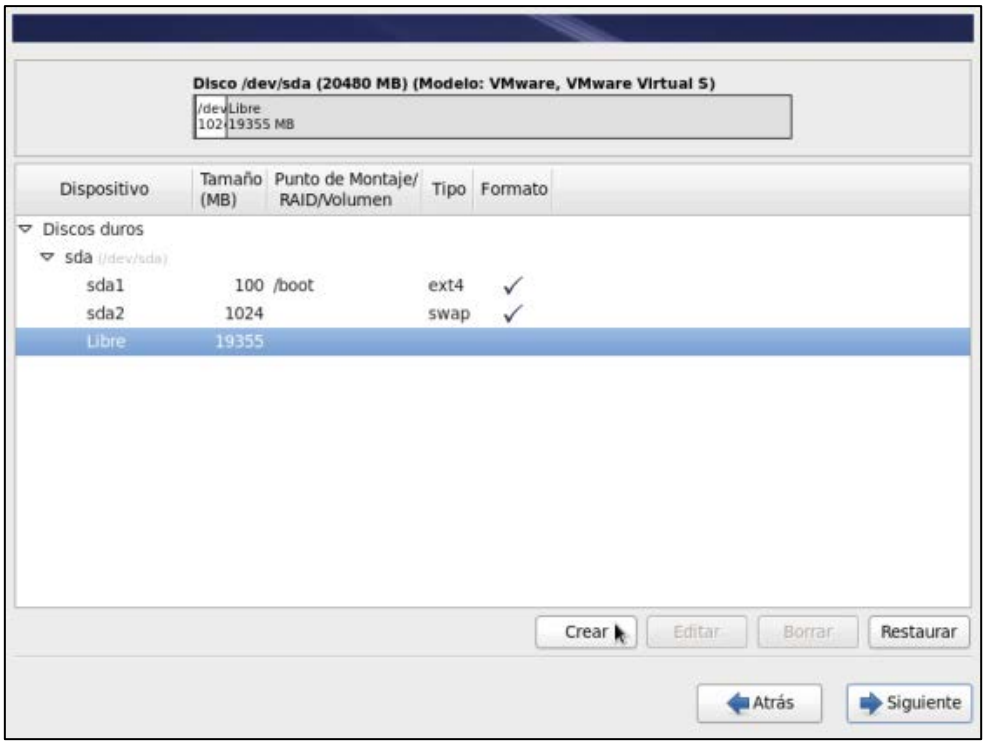
Luego seleccionar Partición estándar y dar click en “Crear”

En el cuadro Añadir partición seleccionar el tipo de sistema de archivos swap con 1024 MB de tamaño fijo.



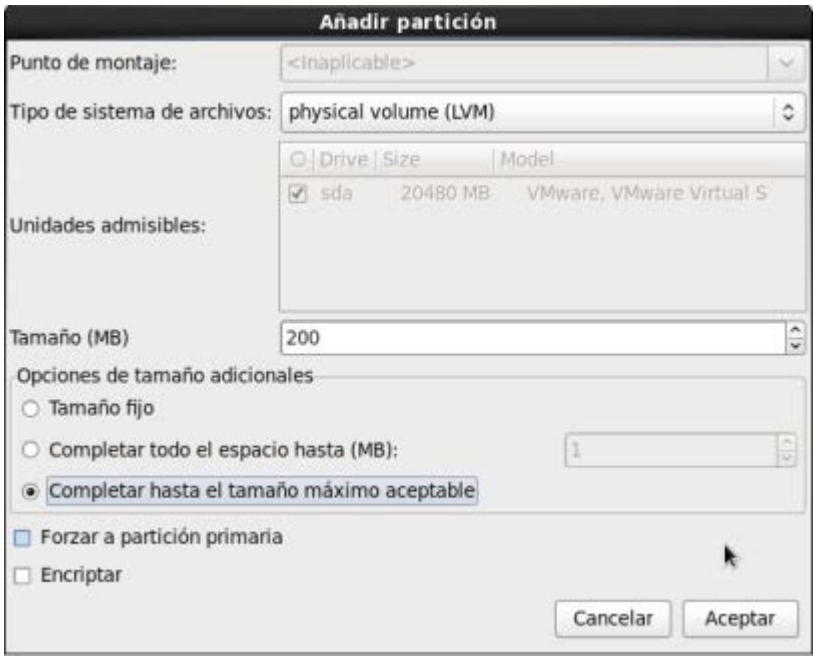
Paso 15 – Crear la partición física LVM

Seleccionar el espacio libre y presionar el botón “Crear”



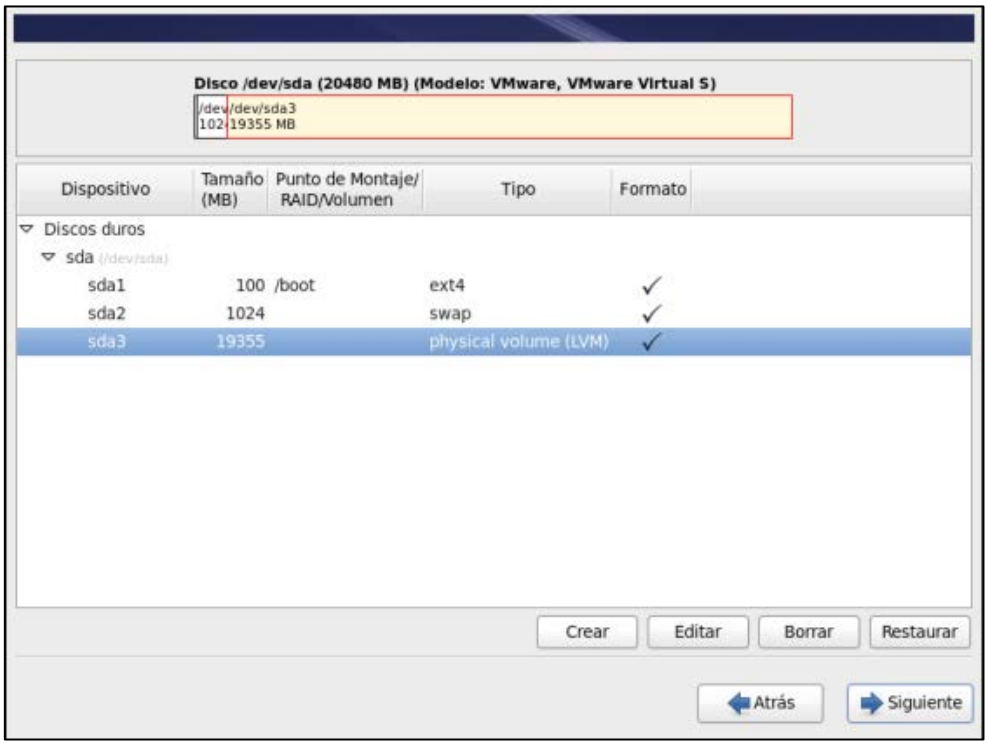
Luego seleccionar Partición estándar y dar click en “Crear”

En el cuadro Añadir partición seleccionar el tipo de sistema de archivos physical volumen (LVM) y la opción “Completar hasta el tamaño máximo aceptable”.

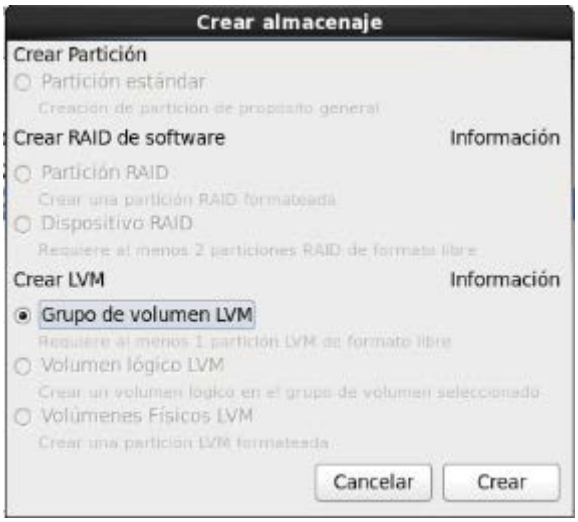


Paso 16 – Crear las particiones lógicas LVM

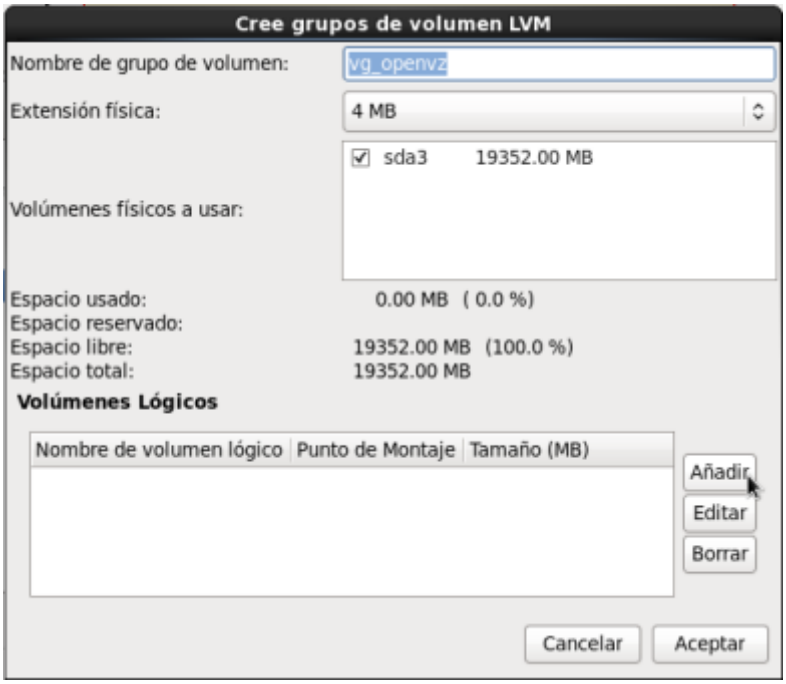
Seleccionar el dispositivo sda3 y dar click en el botón “Crear”



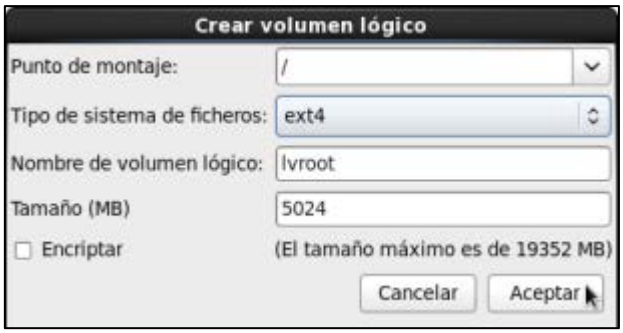
Luego seleccionar “Grupo de volumen LVM”



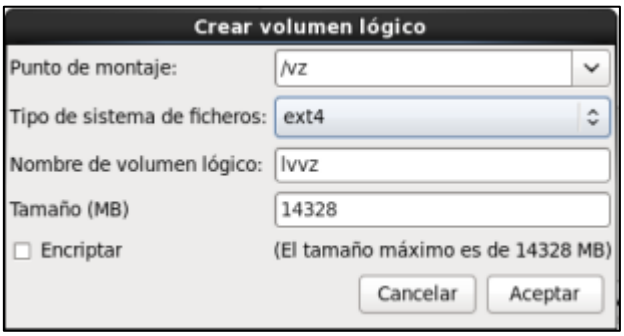
Aparecerá un cuadro con el grupo de volumen llamado vg_openvz, para crear volúmenes lógicos se debe elegir la opción “Añadir”



El primer volumen lógico se llamará lvroot y tendrá el punto de montaje / con tipo de sistema de archivos ext4 y de 5024 MB de tamaño



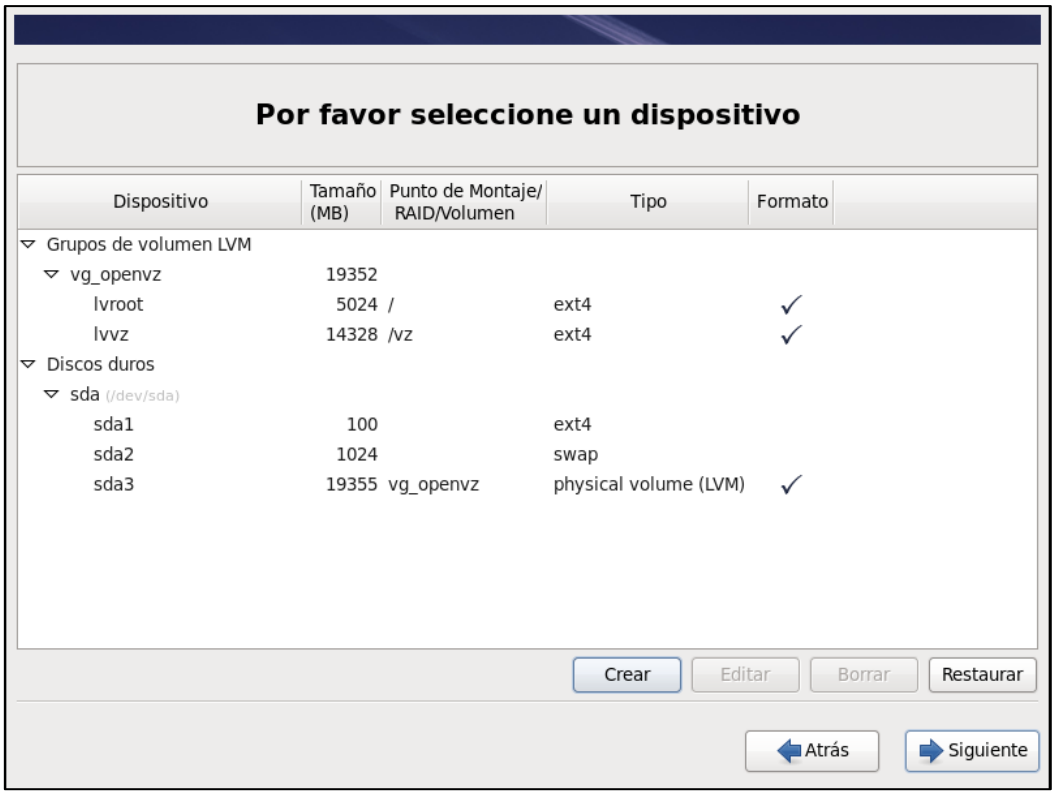
El segundo volumen lógico se llamará lvvz y tendrá el punto de montaje /vz con tipo de sistema de archivos ext4 y de 14328 MB de tamaño



Una vez configurado los volúmenes, se debe elegir la opción “Aceptar”

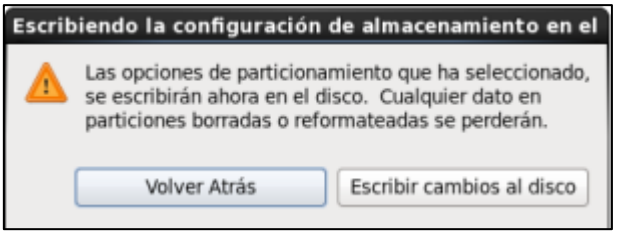
Paso 17 – Formatear el disco

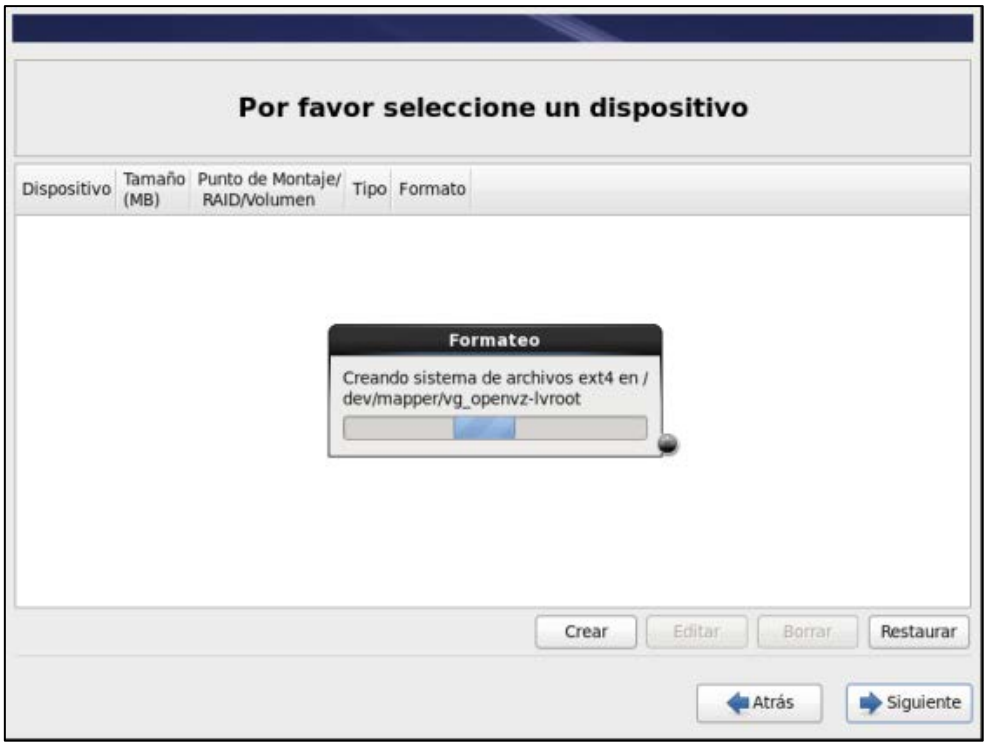
Finalmente, se da click en el botón “Siguiente” para iniciar el formateo



Aparecerá un aviso de formateo al cual se debe presionar “Formato”

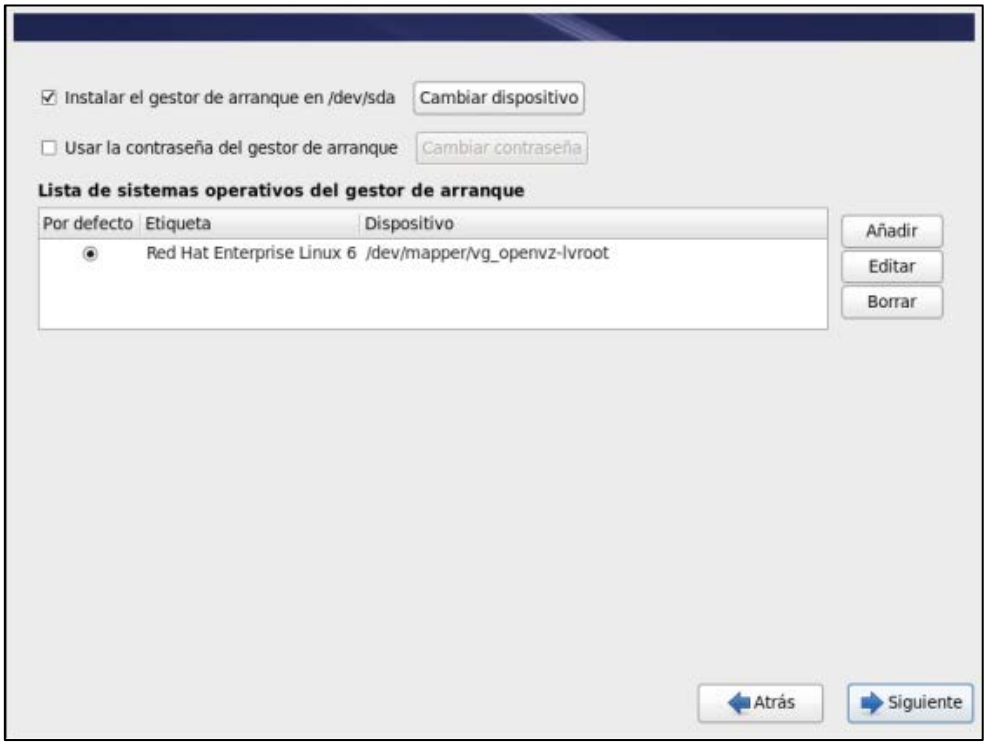
Aparecerá un segundo aviso para confirmar los cambios, para lo que se presionar el botón “Escribir cambios al disco” y el proceso de formato se iniciará



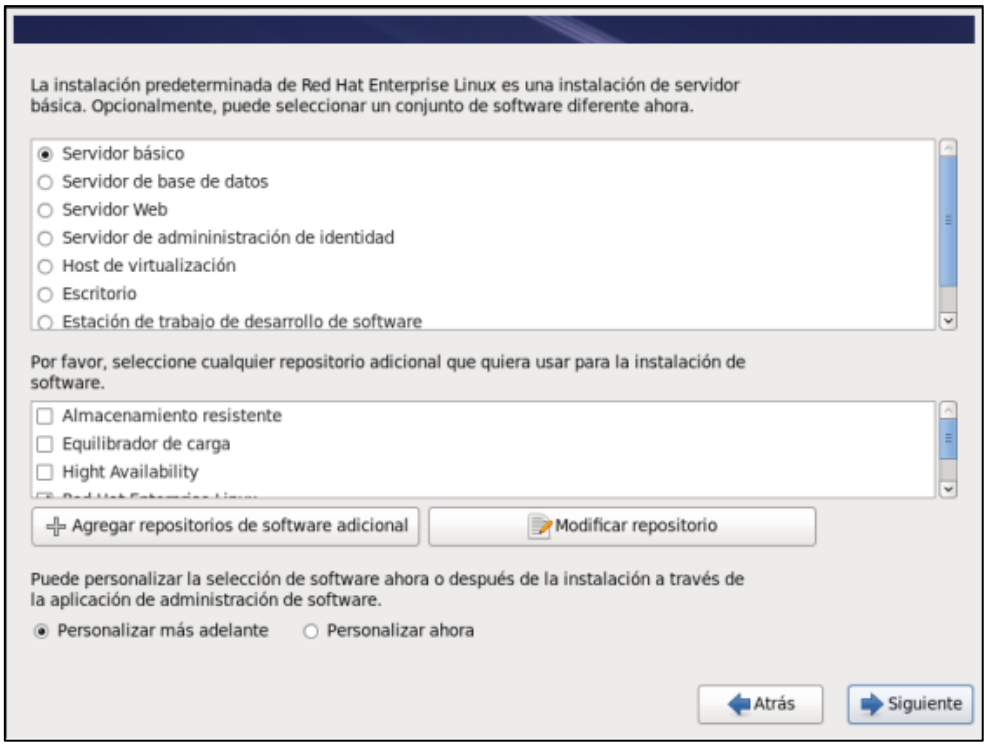


Paso 18 – Configurar el gestor de arranque

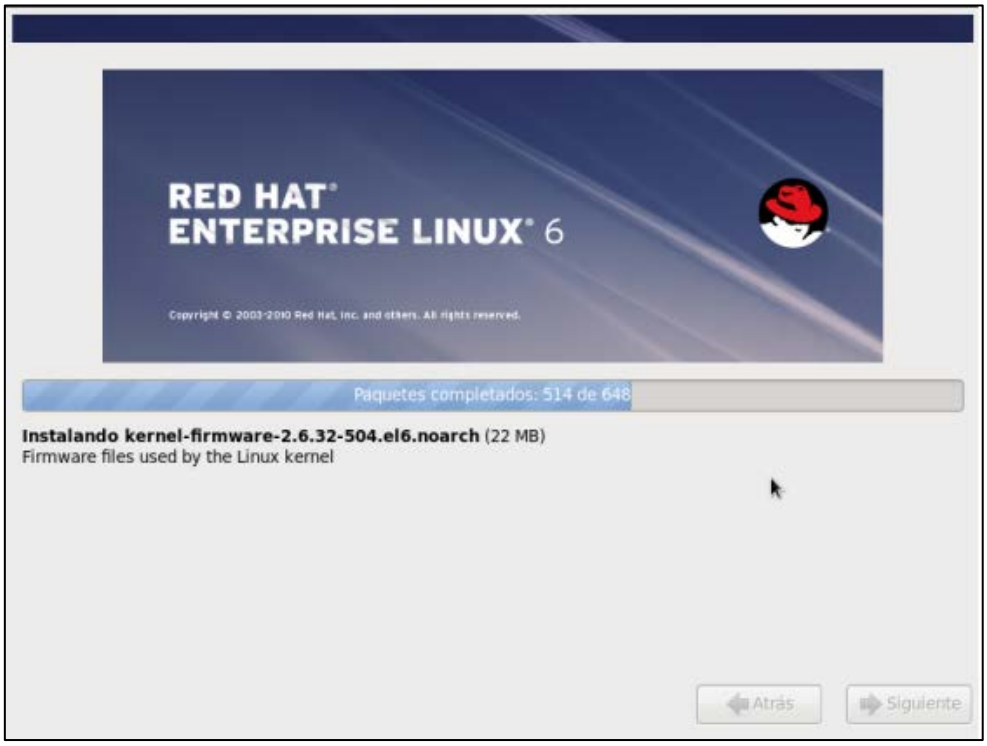
En cuadro para instalar el gestor de arranque, se deja las opciones por defecto



Paso 19 – Elegir la instalación básica



El proceso de instalación se ha iniciado y debe esperarse hasta que termine.



Paso 20 – Reiniciar el servidor

El proceso de instalación ha terminado sin problemas, dar click en “Reiniciar”

B – INSTALACION DE OPENVZ EN RHEL 6

Paso 01 – Aplicar kernel updates

Ingresar como usuario root y ejecutar los siguientes comandos (nota: los kernel updates fueron colocados previamente en la carpeta /root/rhel6.6-updates)

```
cd /root/rhel6.6-updates/
rpm -Uvh kernel-firmware-2.6.32-504.3.3.el6.noarch.rpm
rpm -Uvh kernel-2.6.32-504.3.3.el6.x86_64.rpm
rpm -Uvh kernel-headers-2.6.32-504.3.3.el6.x86_64.rpm
rpm -Uvh kernel-devel-2.6.32-504.3.3.el6.x86_64.rpm
init 6
```

Paso 02 – Configurar repositorio local

Ingresar como usuario root y ejecutar los siguientes comandos para configurar un repositorio local. Como prueba se instalarán los programas vim y wget

```
cd /etc/yum.repos.d
cat > /etc/yum.repos.d/dvd.repo <<EOF
[dvd]
name=Red Hat Enterprise Linux \$releasever - \$basearch -
Source
baseurl=file:///media
enabled=1
gpgcheck=0
EOF
```

```
mount -t iso9660 /dev/cdrom /media/
yum clean all
yum install vim wget
```

Paso 03 – Instalar paquetes openvz

Ingresar como usuario root y ejecutar los siguientes comandos para configurar el repositorio openvz y realizar la instalación de los paquetes necesarios.

```
wget http://download.openvz.org/openvz.repo
rpm --import http://download.openvz.org/RPM-GPG-Key-OpenVZ
yum install vzkernel.x86_64 vzctl vzquota ploop rsync
uname -a
cat /etc/redhat-release
init 6
```

Nota: Los archivos openvz.repo y RPM-GPG-Key-OpenVZ también pueden encontrarse en el CD dentro de la carpeta “recursos\openvz”

Paso 04 – Configurar enviroment

Ingresar como usuario root y ejecutar los siguientes comandos para configurar el enviroment necesario para el funcionamiento de OpenVZ

```
cat > /etc/modprobe.d/openvz.conf <<EOF
options nf_conntrack ip_conntrack_disable_ve0=1
EOF
setenforce 0
cat >> /etc/sysctl.conf <<EOF
#OpenVZ
net.ipv4.ip_forward = 1
net.ipv4.conf.default.proxy_arp = 0
net.ipv4.conf.all.rp_filter = 1
kernel.sysrq = 1
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.all.send_redirects = 0
net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.conf.default.forwarding = 1
EOF
init 6
```

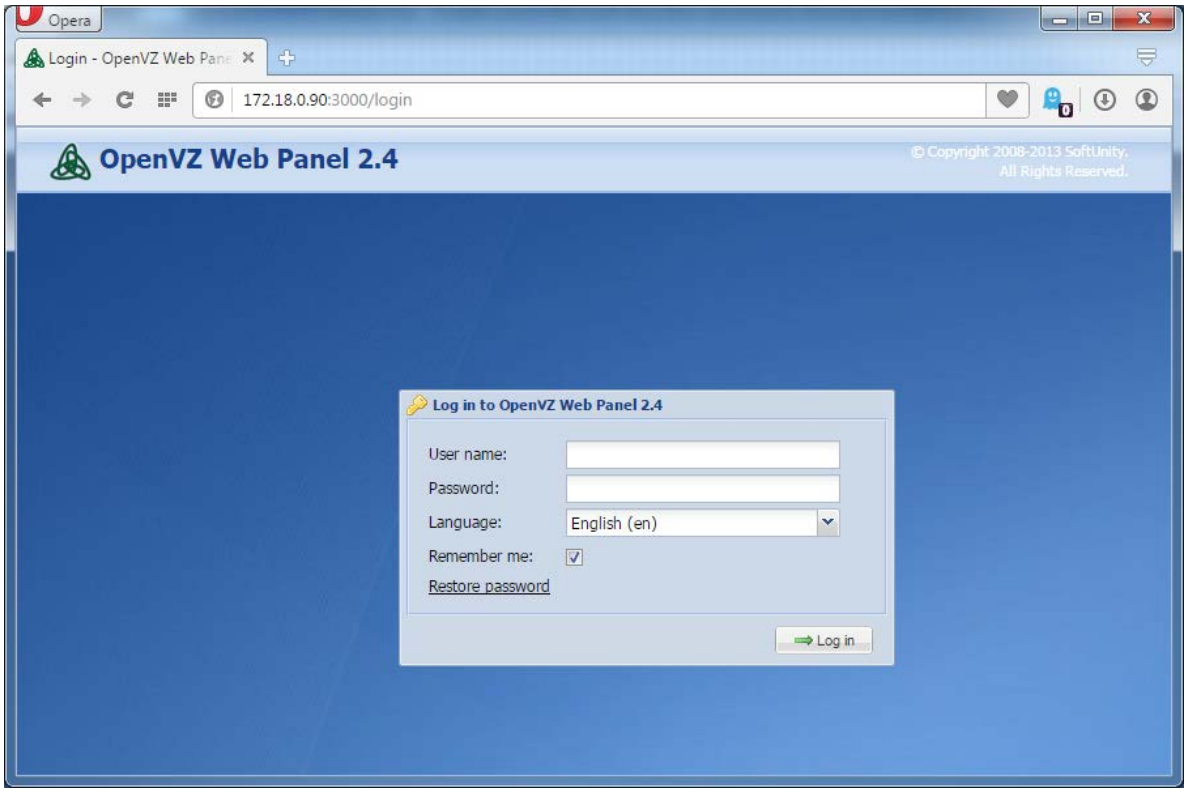
Paso 05 – Instalar OVZ Web Panel

Ingresar como usuario root y ejecutar los siguientes comandos para instalar el panel web OVZ

```
mount -t iso9660 /dev/cdrom /media/
wget -O -
https://raw.githubusercontent.com/sibprogrammer/owp/master/in
staller/ai.sh | sh
```

Nota: El archivo owp-master.zip (que corresponde con el software OpenVZ Web Panel) también pueden encontrarse en el CD dentro de la carpeta “recursos\ openvz”

Revisar la página de inicio de OVZ ingresando a http://172.18.0.90:3000



Anexo 07 - CREACION DE MAQUINAS VIRTUALES

Descripción

Este documento contiene las instrucciones para instalar y configurar las máquinas virtuales correspondientes a los servidores de muestra en la plataforma OpenVZ

Requerimientos Mínimos

- ✓ Plataforma OpenVZ instalado previamente
- ✓ Imagen del sistema operativo Red Hat Enterprise Linux 6

Paso 01 – Crear una nueva máquina virtual Red Hat Enterprise Linux 6

Crear una nueva máquina virtual con las siguientes características:

- ❖ El nombre de la máquina virtual será **rhel6**
- ❖ El nombre del File Disk será **rhel6.vmdk**

Instalar un sistema operativo Red Hat Enterprise Linux 6 en la nueva máquina virtual **rhel6** con las siguientes característica:

- ❖ El nombre del host será **rhel6**

Una vez finalizado el proceso de instalación, se debe obtener el IP del nuevo servidor ejecutando los siguientes comandos con el usuario root

```
ifconfig | grep "inet addr"
```

La finalidad de instalar un nuevo servidor Red Hat Enterprise Linux, es tener una copia sin alteración de un sistema red hat el cual se empleará para la creación de una plantilla en la plataforma OpenVZ.

Paso 02 – Aplicar kernel updates

Ingresar como usuario root al servidor rhel6 y ejecutar los siguientes comandos

```
cd /root/rhel6.6-updates/
rpm -e dracut-kernel-004-356.el6.noarch
rpm -Uvh dracut-004-409.el6_8.2.noarch.rpm
rpm -Uvh dracut-kernel-004-409.el6_8.2.noarch.rpm
rpm -Uvh kernel-2.6.32-696.23.1.el6.x86_64.rpm
rpm -Uvh kernel-headers-2.6.32-696.23.1.el6.x86_64.rpm
rpm -Uvh kernel-devel-2.6.32-696.23.1.el6.x86_64.rpm
init 6
```

Nota: Los archivos de la carpeta rhel6.6-updates fueron colocados previamente en la siguiente ruta del servidor openvz: /root/rhel6.6-updates.

Nota: Los archivos pueden ser encontrados en el CD dentro la carpeta “recursos\ rhel6.6-updates”.

Nota: Para versiones futuras de redhat, se debe revisar el procedimiento de actualización de kernel en la documentación oficial del producto Red Hat Enterprise Linux.

Paso 03 – Transferir el servidor Red Hat a un container OpenVZ

Ingresar como usuario root al servidor **openvz** y ejecutar los siguientes comandos para transferir los archivos del servidor rhel6 hacia el servidor openvz (nota: los archivos serán transferidos al directorio /vz/private/105)

```
mount -t iso9660 /dev/cdrom /media
yum install openssh-clients.x86_64 rsync-3.0.6-12.el6.x86_64
rsync -arvpz --exclude=/dev --exclude=/mnt --exclude=/proc --
exclude=/sys --exclude=/tmp -e ssh root@<ip_servidor_rhel6>:/
/vz/private/105/
```

Donde <ip_servidor_rhel6> corresponde a la dirección IP del servidor rhel6

Nota: El paquete rsync debe estar instalado en ambos servidores

Paso 04 – Modificar los archivos transferidos

Ingresar como usuario root al servidor **openvz** y ejecutar los siguientes comandos

```
sed -i -e 's/^[0-9].*getty.*tty/#&/g'
/vz/private/105/etc/inittab
cat > /vz/private/105/etc/fstab <<EOF
none /dev/pts devpts rw 0 0
EOF
mkdir /vz/private/105/dev
mkdir /vz/private/105/mnt
mkdir /vz/private/105/proc
mkdir /vz/private/105/sys
mkdir /vz/private/105/tmp
mkdir /vz/private/105/dev/pts
mkdir /vz/private/105/etc/udev/devices
/sbin/MAKEDEV -d /vz/private/105/dev -x
{p,t}ty{a,p}{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f} console core
full kmem kmsg mem null port ptmx random urandom zero ram0
/sbin/MAKEDEV -d /vz/private/105/etc/udev/devices -x
{p,t}ty{a,p}{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f} console core
full kmem kmsg mem null port ptmx random urandom zero ram0
chmod 1777 /vz/private/105/tmp
```

```
chmod 1777 /vz/private/105/var/tmp
sed -i 's/^\(ONBOOT=\).*/\1NO/'
/vz/private/105/etc/sysconfig/network-scripts/ifcfg-eth0
sed -i 's/console output/#console output/g'
/vz/private/105/etc/init/rc.conf
sed -i 's/console output/#console output/g'
/vz/private/105/etc/init/rcS.conf
sed -i 's/console output/#console output/g'
/vz/private/105/etc/init/tty.conf
mv /vz/private/105/etc/init/tty.conf
/vz/private/105/etc/init/tty.conf.disable
mv /vz/private/105/etc/init/start-ttys.conf
/vz/private/105/etc/init/start-ttys.conf.disable
cd /vz/private/105
tar -cvzf /vz/template/cache/rhel-6-x86_64.tar.gz ./
cd
rm -Rf /vz/private/105
```

Paso 05 – Crear las máquinas virtuales

Ingresar como usuario root al servidor **openvz** y ejecutar los siguientes comandos

```
vzctl create 105 --ostemplate rhel-6-x86_64 --config basic
vzctl set 105 --onboot yes --save
vzctl set 105 --hostname vz-central --save
vzctl set 105 --ipadd 172.18.0.81 --save
vzctl set 105 --nameserver 8.8.8.8 --nameserver 8.8.4.4 --
save
vzctl set 105 --userpasswd root:miclave
vzctl start 105
vzlist –a

vzctl create 106 --ostemplate rhel-6-x86_64 --config basic
vzctl set 106 --onboot yes --save
vzctl set 106 --hostname vz-proxy --save
vzctl set 106 --ipadd 172.18.0.82 --save
vzctl set 106 --nameserver 8.8.8.8 --nameserver 8.8.4.4 --
save
vzctl set 106 --userpasswd root:miclave
vzctl start 106
vzlist -a

vzctl create 107 --ostemplate rhel-6-x86_64 --config basic
vzctl set 107 --onboot yes --save
vzctl set 107 --hostname vz-mail --save
vzctl set 107 --ipadd 172.18.0.83 --save
vzctl set 107 --nameserver 8.8.8.8 --nameserver 8.8.4.4 –save
```

```
vzctl set 107 --userpasswd root:miclave
vzctl start 107
vzlist -a

vzctl create 108 --ostemplate rhel-6-x86_64 --config basic
vzctl set 108 --onboot yes --save
vzctl set 108 --hostname vz-file --save
vzctl set 108 --ipadd 172.18.0.84 --save
vzctl set 108 --nameserver 8.8.8.8 --nameserver 8.8.4.4 –save
vzctl set 108 --userpasswd root:miclave
vzctl start 108
vzlist -a

vzctl create 109 --ostemplate rhel-6-x86_64 --config basic
vzctl set 109 --onboot yes --save
vzctl set 109 --hostname vz-backup --save
vzctl set 109 --ipadd 172.18.0.85 --save
vzctl set 109 --nameserver 8.8.8.8 --nameserver 8.8.4.4 –save
vzctl set 109 --userpasswd root:miclave
vzctl start 109
vzlist -a

vzctl create 110 --ostemplate rhel-6-x86_64 --config basic
vzctl set 110 --onboot yes --save
vzctl set 110 --hostname vz-syslog --save
vzctl set 110 --ipadd 172.18.0.86 --save
vzctl set 110 --nameserver 8.8.8.8 --nameserver 8.8.4.4 –save
vzctl set 110 --userpasswd root:miclave
vzctl start 110
vzlist -a

vzctl create 111 --ostemplate rhel-6-x86_64 --config basic
vzctl set 111 --onboot yes --save
vzctl set 111 --hostname vz-directory --save
vzctl set 111 --ipadd 172.18.0.87 --save
vzctl set 111 --nameserver 8.8.8.8 --nameserver 8.8.4.4 –save
vzctl set 111 --userpasswd root:miclave
vzctl start 111
vzlist -a

vzctl create 112 --ostemplate rhel-6-x86_64 --config basic
vzctl set 112 --onboot yes --save
vzctl set 112 --hostname vz-monitor01 --save
vzctl set 112 --ipadd 172.18.0.88 --save
vzctl set 112 --nameserver 8.8.8.8 --nameserver 8.8.4.4 –save
vzctl set 112 --userpasswd root:miclave
vzctl start 112
```

```
vzlist -a

vzctl create 113 --ostemplate rhel-6-x86_64 --config basic
vzctl set 113 --onboot yes --save
vzctl set 113 --hostname vz-monitor02 --save
vzctl set 113 --ipadd 172.18.0.89 --save
vzctl set 113 --nameserver 8.8.8.8 --nameserver 8.8.4.4 –save
vzctl set 113 --userpasswd root:miclave
vzctl start 113
vzlist –a
```

Nota: Por consideraciones de seguridad, las claves aquí mostradas no deben ser utilizadas durante este procedimiento.

Paso 06 – Configurar entradas de filesystem

Ingresar como usuario root al servidor **openvz** y ejecutar los siguientes comandos

```
vzctl stop 105
vzctl mount 105
cd /vz/root/105/etc/
rm mtab
ln -s /proc/mounts mtab
cd
vzctl umount 105
vzctl start 105

vzctl stop 106
vzctl mount 106
cd /vz/root/106/etc/
rm mtab
ln -s /proc/mounts mtab
cd
vzctl umount 106
vzctl start 106

vzctl stop 107
vzctl mount 107
cd /vz/root/107/etc/
rm mtab
ln -s /proc/mounts mtab
cd
vzctl umount 107
vzctl start 107

vzctl stop 108
vzctl mount 108
cd /vz/root/108/etc/
rm mtab
ln -s /proc/mounts mtab
cd
vzctl umount 108
vzctl start 108

vzctl stop 109
vzctl mount 109
cd /vz/root/109/etc/
rm mtab
ln -s /proc/mounts mtab
cd
vzctl umount 109
vzctl start 109
```

```
vzctl stop 110
vzctl mount 110
cd /vz/root/110/etc/
rm mtab
ln -s /proc/mounts mtab
cd
vzctl umount 110
vzctl start 110

vzctl stop 111
vzctl mount 111
cd /vz/root/111/etc/
rm mtab
ln -s /proc/mounts mtab
cd
vzctl umount 111
vzctl start 111

vzctl stop 112
vzctl mount 112
cd /vz/root/112/etc/
rm mtab
ln -s /proc/mounts mtab
cd
vzctl umount 112
vzctl start 112

vzctl stop 113
vzctl mount 113
cd /vz/root/113/etc/
rm mtab
ln -s /proc/mounts mtab
cd
vzctl umount 113
vzctl start 113
```

Estos comandos tienen la finalidad de evitar errores en el reconocimiento del filesystem por parte del contenedor.

Anexo 08 - CONFIGURACION DE LAS MAQUINAS VIRTUALES

Descripción

Este documento contiene las instrucciones para instalar y configurar los servicios que debería tener cada servidor según la muestra del proyecto y de esta manera tener un entorno de simulación para el protocolo sump

Requerimientos Mínimos

- ✓ Plataforma OpenVZ instalado previamente
- ✓ Imagen del sistema operativo Red Hat Enterprise Linux 6
- ✓ Información de usuarios y grupos por cada sistema o servicio a configurar

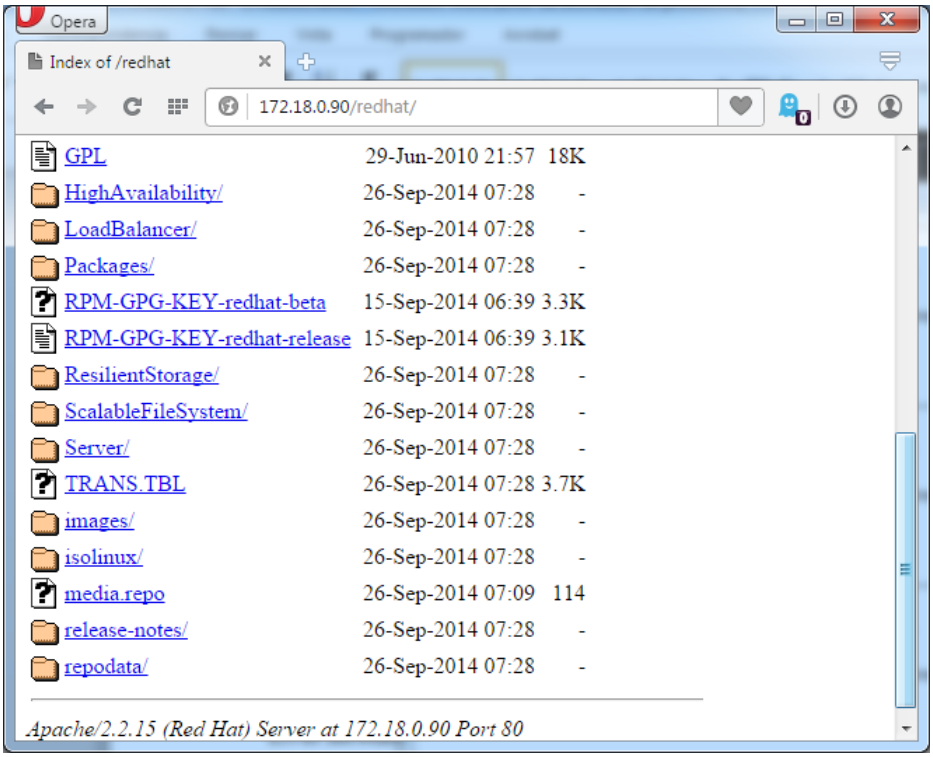
Paso 01 – Configurar un repositorio web

En el servidor **openvz**, ejecutar los siguientes comandos con usuario root

```
mount -t iso9660 /dev/cdrom /media/
yum install httpd
chkconfig httpd on
service httpd start
service iptables stop
cd /var/www/html
ln -s /media redhat
```

Para comprobar, acceder a un navegador y verificar el funcionamiento del repositorio.

En el caso de este proyecto, el servidor openvz tiene la IP 172.18.0.90 por lo tanto, la URL a acceder es: http://172.18.0.90/redhat/



Paso 02 – Configurar los repositorios locales en las máquinas virtuales

En cada servidor de la simulación, ejecutar los siguientes comandos con usuario root

```
echo “172.18.0.90 openvz” >> /etc/hosts
cat > /etc/yum.repos.d/dvd.repo <<EOF
[dvd]
name=Red Hat Enterprise Linux \$releasever - \$basearch -
Source
baseurl=http://openvz/redhat
enabled=1
gpgcheck=0
EOF
```

Nota: Esta configuración debe realizarse entrando de forma remota (ssh) a cada servidor creado (a excepción de openvz). El IP 172.18.0.90 corresponde al número IP del servidor openvz.

Paso 03 – Configurar el servidor vz-central

En el servidor vz-central, ejecutar los siguientes comandos con usuario root

```
yum clean all
yum install gcc gdb openssh-clients
```

```
yum install mysql.x86_64 mysql-devel.x86_64 mysql-libs.x86_64
mysql-server.x86_64
chkconfig mysqld on
service mysqld start
mysql_secure_installation
```

Nota: al ejecutar mysql_secure_installation, responder las siguientes preguntas

```
Set root password? [Y/n] Y
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y
```

Luego, se procede a crear los usuarios linux de prueba

```
groupadd administradores
cat /etc/group
adduser -d /home/admin01 -c "Administrador Petroperú 01" -g
administradores admin01
adduser -d /home/admin02 -c "Administrador Petroperú 02" -g
administradores admin02
adduser -d /home/admin03 -c "Administrador Petroperú 03" -g
administradores admin03
adduser -d /home/admin04 -c "Administrador Petroperú 04" -g
administradores admin04
adduser -d /home/admin05 -c "Administrador Petroperú 05" -g
administradores admin05
groupadd onsites
cat /etc/group
adduser -d /home/onsite01 -c "Onsite Diurno" -g onsites
onsite01
adduser -d /home/onsite02 -c "Onsite Nocturno" -g onsites
onsite02
passwd admin01
passwd admin02
passwd admin03
passwd admin04
passwd admin05
passwd onsite01
passwd onsite02
```

Después, se crea el archivo script.sql (usando el comando: vim script.sql) con el siguiente contenido

```
/*!40000 DROP DATABASE IF EXISTS `sistema_x`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_x` /*!40100
DEFAULT CHARACTER SET latin1 */;
```

```
USE `sistema_x`;

/*!40000 DROP DATABASE IF EXISTS `sistema_y`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_y` /*!40100
DEFAULT CHARACTER SET latin1 */;
USE `sistema_y`;

/*!40000 DROP DATABASE IF EXISTS `sistema_z`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_z` /*!40100
DEFAULT CHARACTER SET latin1 */;
USE `sistema_z`;

/*!40000 DROP DATABASE IF EXISTS `sistema_p`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_p` /*!40100
DEFAULT CHARACTER SET latin1 */;
USE `sistema_p`;

GRANT ALL PRIVILEGES ON *.* TO 'dba'@'localhost' IDENTIFIED
BY 'clavedba';
GRANT ALL PRIVILEGES ON sistema_x.* TO 'sis_x'@'localhost'
IDENTIFIED BY 'clave';
GRANT ALL PRIVILEGES ON sistema_y.* TO 'sis_y'@'localhost'
IDENTIFIED BY 'clave';
GRANT ALL PRIVILEGES ON sistema_z.* TO 'sis_z'@'localhost'
IDENTIFIED BY 'clave';
GRANT ALL PRIVILEGES ON sistema_p.* TO 'sis_p'@'localhost'
IDENTIFIED BY 'clave';
GRANT SELECT ON mysql.* TO 'sump'@'localhost' IDENTIFIED BY
'clave-sump';
FLUSH PRIVILEGES;
```

Nota: Por consideraciones de seguridad, todas las claves aquí descritas se deben cambiar durante este proceso.

Finalmente, se ejecuta el script sql para crear los usuarios mysql para la prueba

```
mysql -u root -p < script.sql
```

Paso 04 – Configurar el servidor vz-proxy

En el servidor vz-proxy, ejecutar los siguientes comandos con usuario root

```
yum clean all
yum install gcc gdb openssh-clients
```

Luego, se procede a crear los usuarios linux de prueba

```
groupadd administradores
cat /etc/group
adduser -d /home/admin01 -c "Administrador Proxy Petroperú
01" -g administradores admin01
adduser -d /home/admin02 -c "Administrador Proxy Petroperú
02" -g administradores admin02
adduser -d /home/admin03 -c "Administrador Proxy Petroperú
03" -g administradores admin03
adduser -d /home/admin04 -c "Administrador Proxy Petroperú
04" -g administradores admin04
adduser -d /home/admin05 -c "Administrador Proxy Petroperú
05" -g administradores admin05
passwd admin01
passwd admin02
passwd admin03
passwd admin04
passwd admin05
```

Paso 05 – Configurar el servidor vz-mail

En el servidor vz-mail, ejecutar los siguientes comandos con usuario root

```
yum clean all
yum install gcc gdb openssh-clients
```

Luego, se procede a crear los usuarios linux de prueba

```
groupadd administradores
cat /etc/group
adduser -d /home/admin01 -c "Administrador Mail Petroperú 01"
-g administradores admin01
adduser -d /home/admin02 -c "Administrador Mail Petroperú 02"
-g administradores admin02
adduser -d /home/admin03 -c "Administrador Mail Petroperú 03"
-g administradores admin03
adduser -d /home/admin04 -c "Administrador Mail Petroperú 04"
-g administradores admin04
adduser -d /home/admin05 -c "Administrador Mail Petroperú 05"
-g administradores admin05
passwd admin01
passwd admin02
passwd admin03
passwd admin04
passwd admin05
```

Paso 06 – Configurar el servidor vz-file

En el servidor vz-file, ejecutar los siguientes comandos con usuario root

```
yum clean all
yum install gcc gdb openssh-clients
```

Luego, se procede a crear los usuarios linux de prueba

```
groupadd administradores
cat /etc/group
adduser -d /home/admin01 -c "Administrador File Petroperú 01"
-g administradores admin01
adduser -d /home/admin02 -c "Administrador File Petroperú 02"
-g administradores admin02
adduser -d /home/admin03 -c "Administrador File Petroperú 03"
-g administradores admin03
adduser -d /home/admin04 -c "Administrador File Petroperú 04"
-g administradores admin04
adduser -d /home/admin05 -c "Administrador File Petroperú 05"
-g administradores admin05
passwd admin01
passwd admin02
passwd admin03
passwd admin04
passwd admin05
```

Paso 07 – Configurar el servidor vz-backup

En el servidor vz-backup, ejecutar los siguientes comandos con usuario root

```
yum clean all
yum install gcc gdb openssh-clients
```

Luego, se procede a crear los usuarios linux de prueba

```
groupadd administradores
cat /etc/group
adduser -d /home/admin01 -c "Administrador Backup Petroperú
01" -g administradores admin01
adduser -d /home/admin02 -c "Administrador Backup Petroperú
02" -g administradores admin02
adduser -d /home/admin03 -c "Administrador Backup Petroperú
03" -g administradores admin03
adduser -d /home/admin04 -c "Administrador Backup Petroperú
04" -g administradores admin04
adduser -d /home/admin05 -c "Administrador Backup Petroperú
05" -g administradores admin05
passwd admin01
passwd admin02
```

```
passwd admin03
passwd admin04
passwd admin05
```

Paso 08 – Configurar el servidor vz-syslog

En el servidor vz-syslog, ejecutar los siguientes comandos con usuario root

```
yum clean all
yum install gcc gdb openssh-clients
```

Luego, se procede a crear los usuarios linux de prueba

```
groupadd administradores
cat /etc/group
adduser -d /home/admin01 -c "Administrador Syslog Petroperú
01" -g administradores admin01
adduser -d /home/admin02 -c "Administrador Syslog Petroperú
02" -g administradores admin02
adduser -d /home/admin03 -c "Administrador Syslog Petroperú
03" -g administradores admin03
adduser -d /home/admin04 -c "Administrador Syslog Petroperú
04" -g administradores admin04
adduser -d /home/admin05 -c "Administrador Syslog Petroperú
05" -g administradores admin05
passwd admin01
passwd admin02
passwd admin03
passwd admin04
passwd admin05
```

Paso 09 – Configurar el servidor vz-directory

En el servidor vz-directory, ejecutar los siguientes comandos con usuario root

```
yum clean all
yum install gcc gdb openssh-clients
```

Luego, se procede a crear los usuarios linux de prueba

```
groupadd administradores
cat /etc/group
adduser -d /home/admin01 -c "Administrador Directory
Petroperú 01" -g administradores admin01
adduser -d /home/admin02 -c "Administrador Directory
Petroperú 02" -g administradores admin02
adduser -d /home/admin03 -c "Administrador Directory
Petroperú 03" -g administradores admin03
```

```
adduser -d /home/admin04 -c "Administrador Directory
Petroperú 04" -g administradores admin04
adduser -d /home/admin05 -c "Administrador Directory
Petroperú 05" -g administradores admin05
passwd admin01
passwd admin02
passwd admin03
passwd admin04
passwd admin05
```

Paso 10 – Configurar el servidor vz-monitor01

En el servidor vz-monitor01, ejecutar los siguientes comandos con usuario root

```
yum clean all
yum install gcc gdb openssh-clients
yum install mysql.x86_64 mysql-devel.x86_64 mysql-libs.x86_64
mysql-server.x86_64
chkconfig mysqld on
service mysqld start
mysql_secure_installation
```

Nota: al ejecutar mysql_secure_installation, responder las siguientes preguntas

```
Set root password? [Y/n] Y
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y
```

Luego, se procede a crear los usuarios linux de prueba

```
groupadd administradores
cat /etc/group
adduser -d /home/admin01 -c "Administrador Monitor01
Petroperú 01" -g administradores admin01
adduser -d /home/admin02 -c "Administrador Monitor01
Petroperú 02" -g administradores admin02
adduser -d /home/admin03 -c "Administrador Monitor01
Petroperú 03" -g administradores admin03
adduser -d /home/admin04 -c "Administrador Monitor01
Petroperú 04" -g administradores admin04
adduser -d /home/admin05 -c "Administrador Monitor01
Petroperú 05" -g administradores admin05
groupadd onsites
cat /etc/group
adduser -d /home/onsite01 -c "Onsite Monitor01 Diurno" -g
onsites onsite01
```

```
adduser -d /home/onsite02 -c "Onsite Monitor01 Nocturno" -g
onsites onsite02
passwd admin01
passwd admin02
passwd admin03
passwd admin04
passwd admin05
passwd onsite01
passwd onsite02
```

Después, se crea el archivo script.sql (usando el comando: vim script.sql) con el siguiente contenido

```
/*!40000 DROP DATABASE IF EXISTS `sistema_a`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_a` /*!40100
DEFAULT CHARACTER SET latin1 */;
USE `sistema_a`;

/*!40000 DROP DATABASE IF EXISTS `sistema_b`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_b` /*!40100
DEFAULT CHARACTER SET latin1 */;
USE `sistema_b`;

/*!40000 DROP DATABASE IF EXISTS `sistema_c`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_c` /*!40100
DEFAULT CHARACTER SET latin1 */;
USE `sistema_c`;

/*!40000 DROP DATABASE IF EXISTS `sistema_d`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_d` /*!40100
DEFAULT CHARACTER SET latin1 */;
USE `sistema_d`;

GRANT ALL PRIVILEGES ON *.* TO 'dba'@'localhost' IDENTIFIED
BY 'clavedba';
GRANT ALL PRIVILEGES ON sistema_a.* TO 'sis_a'@'localhost'
IDENTIFIED BY 'clave';
GRANT ALL PRIVILEGES ON sistema_b.* TO 'sis_b'@'localhost'
IDENTIFIED BY 'clave';
GRANT ALL PRIVILEGES ON sistema_c.* TO 'sis_c'@'localhost'
IDENTIFIED BY 'clave';
GRANT ALL PRIVILEGES ON sistema_d.* TO 'sis_d'@'localhost'
IDENTIFIED BY 'clave';
GRANT SELECT ON mysql.* TO 'sump'@'localhost' IDENTIFIED BY
'clave-sump';
FLUSH PRIVILEGES;
```

Nota: Por consideraciones de seguridad, todas las claves aquí descritas se deben cambiar durante este proceso.

Finalmente, se ejecuta el script sql para crear los usuarios mysql para la prueba

```
mysql -u root -p < script.sql
```

Paso 11 – Configurar el servidor vz-monitor02

En el servidor vz-monitor02, ejecutar los siguientes comandos con usuario root

```
yum clean all
yum install gcc gdb openssh-clients
yum install mysql.x86_64 mysql-devel.x86_64 mysql-libs.x86_64
mysql-server.x86_64
chkconfig mysqld on
service mysqld start
mysql_secure_installation
```

Nota: al ejecutar mysql_secure_installation, responder las siguientes preguntas

```
Set root password? [Y/n] Y
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y
```

Luego, se procede a crear los usuarios linux de prueba

```
groupadd administradores
cat /etc/group
adduser -d /home/admin01 -c "Administrador Monitor02
Petroperú 01" -g administradores admin01
adduser -d /home/admin02 -c "Administrador Monitor02
Petroperú 02" -g administradores admin02
adduser -d /home/admin03 -c "Administrador Monitor02
Petroperú 03" -g administradores admin03
adduser -d /home/admin04 -c "Administrador Monitor02
Petroperú 04" -g administradores admin04
adduser -d /home/admin05 -c "Administrador Monitor02
Petroperú 05" -g administradores admin05
groupadd onsites
cat /etc/group
adduser -d /home/onsite01 -c "Onsite Monitor02 Diurno" -g
onsites onsite01
adduser -d /home/onsite02 -c "Onsite Monitor02 Nocturno" -g
onsites onsite02
passwd admin01
```

```
passwd admin02
passwd admin03
passwd admin04
passwd admin05
passwd onsite01
passwd onsite02
```

Después, se crea el archivo script.sql (usando el comando: vim script.sql) con el siguiente contenido

```
/*!40000 DROP DATABASE IF EXISTS `sistema_a`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_a` /*!40100
DEFAULT CHARACTER SET latin1 */;
USE `sistema_a`;

/*!40000 DROP DATABASE IF EXISTS `sistema_b`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_b` /*!40100
DEFAULT CHARACTER SET latin1 */;
USE `sistema_b`;

/*!40000 DROP DATABASE IF EXISTS `sistema_c`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_c` /*!40100
DEFAULT CHARACTER SET latin1 */;
USE `sistema_c`;

/*!40000 DROP DATABASE IF EXISTS `sistema_d`*/;
CREATE DATABASE /*!32312 IF NOT EXISTS*/ `sistema_d` /*!40100
DEFAULT CHARACTER SET latin1 */;
USE `sistema_d`;

GRANT ALL PRIVILEGES ON *.* TO 'dba'@'localhost' IDENTIFIED
BY 'clavedba';
GRANT ALL PRIVILEGES ON sistema_a.* TO 'sis_a'@'localhost'
IDENTIFIED BY 'clave';
GRANT ALL PRIVILEGES ON sistema_b.* TO 'sis_b'@'localhost'
IDENTIFIED BY 'clave';
GRANT ALL PRIVILEGES ON sistema_c.* TO 'sis_c'@'localhost'
IDENTIFIED BY 'clave';
GRANT ALL PRIVILEGES ON sistema_d.* TO 'sis_d'@'localhost'
IDENTIFIED BY 'clave';
GRANT SELECT ON mysql.* TO 'sump'@'localhost' IDENTIFIED BY
'clave-sump';
FLUSH PRIVILEGES;
```

Nota: Por consideraciones de seguridad, todas las claves aquí descritas se deben cambiar durante este proceso.

Finalmente, se ejecuta el script sql para crear los usuarios mysql para la prueba

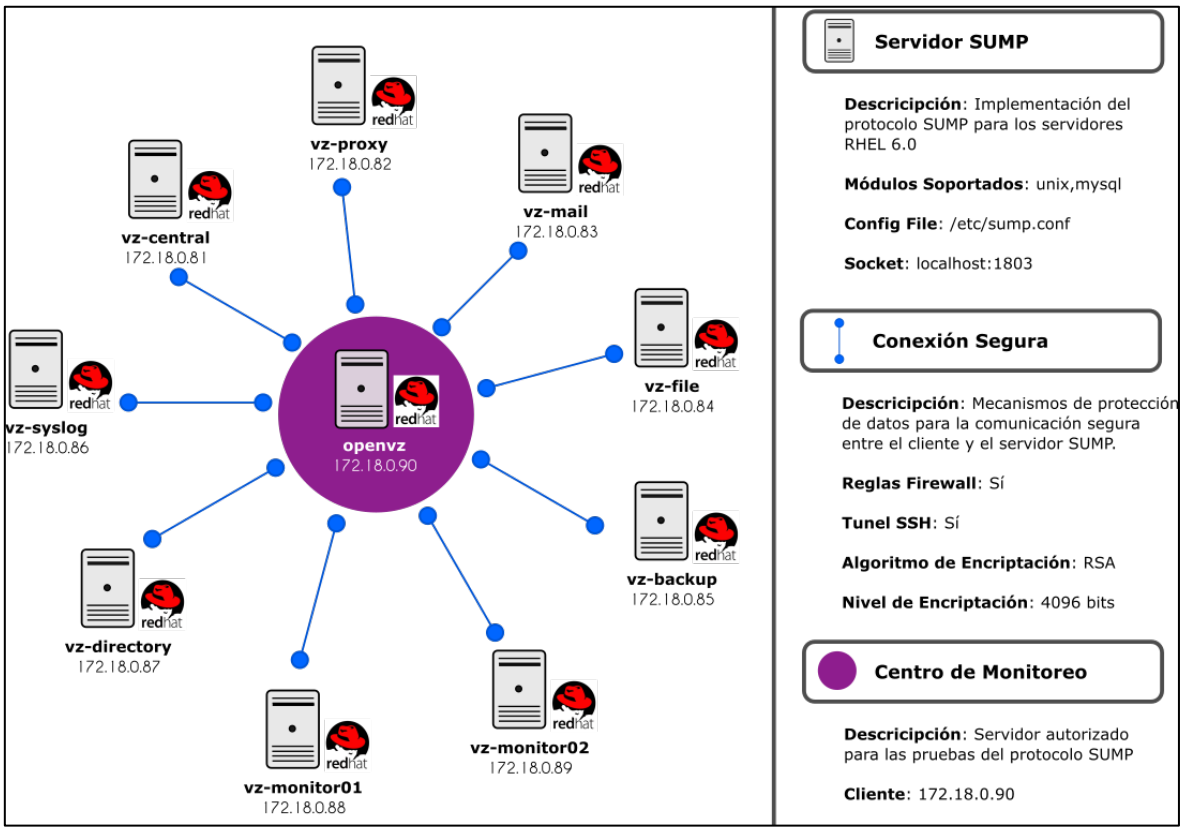
```
mysql -u root -p < script.sql
```

Anexo 09 – ARQUITECTURA DE LA SIMULACION

Descripción

Este documento contiene la arquitectura de la simulación que se utilizará para probar el protocolo sump.

ARQUITECTURA DE LA SIMULACION



Anexo 10 – INSTALACION DEL PROTOCOLO SUMP

Descripción

Este documento contiene las instrucciones para instalar y configurar el protocolo sump en las máquinas virtuales de la red de simulación

Requerimientos Mínimos

- ✓ Máquinas virtuales de simulación creadas y configuradas
- ✓ Software sump previamente copiado al servidor openvz

Paso 01 – Copiar software a las máquinas virtuales

Ejecutar los siguientes comandos con usuario root en el servidor openvz.

```
scp –r /root/sump/ root@172.18.0.81:/root/sumpd
scp –r /root/sump/ root@172.18.0.82:/root/sumpd
scp –r /root/sump/ root@172.18.0.83:/root/sumpd
scp –r /root/sump/ root@172.18.0.84:/root/sumpd
scp –r /root/sump/ root@172.18.0.85:/root/sumpd
scp –r /root/sump/ root@172.18.0.86:/root/sumpd
scp –r /root/sump/ root@172.18.0.87:/root/sumpd
scp –r /root/sump/ root@172.18.0.88:/root/sumpd
scp –r /root/sump/ root@172.18.0.89:/root/sumpd
```

Nota: Se ha colocado previamente el software en la carpeta /root/sump del servidor openvz.

Nota: El código fuente puede conseguirse en el CD del proyecto de Tesis, en la carpeta “software\sump”.

Paso 02 – Copiar el archivo de configuración en cada servidor sump

Ejecutar los siguientes comandos con usuario root cada máquina virtual

```
scp /root/sump/config/config.x.json
root@172.18.0.x:/etc/sump.conf
```

Donde x es el último octeto de los números ip de los servidores.

Anexo 11 – EL ARCHIVO DE CONFIGURACIÓN DEL SERVIDOR

Descripción

Este documento contiene los detalles técnicos del archivo de configuración /etc/sump.conf utilizado por el servidor sump para condicionar el comportamiento del software. El formato utilizado por este archivo de configuración es JSON.

Esquema del archivo /etc/sump.conf

```
{
    “OPCIONES_GENERALES”:”VALOR”, …,
    “modules”: [ CONFIG_MODULES ]
}
```

Dónde:

“OPCIONES_GENERALES”: Corresponden a ciertas opciones que afectarán a todo el programa

“VALOR”: Corresponde a los valores de las opciones generales

“modules”: Corresponde con el inicio de la sección de módulos

[CONFIG_MODULES]: Corresponden a las opciones de los módulos

SECCION OPCIONES GENERALES

La sección general está compuesta por las siguientes opciones:

Opción	Descripción
`hostname`	Nombre del host donde se activara el servicio
`port`	Puerto de red para la comunicación
`user`	Usuario del servicio
`group`	Grupo del servicio
`log`	Archivo log del servicio
`clients`	Lista hash de clientes permitidos
`modules`	Lista de módulos. Debe haber al menos una entrada

SECCION MODULES

La sección del módulo UNIX es la sección utilizada para gestionar el módulo de cuentas de usuario linux y está compuesta por las siguientes opciones:

Opción	Descripción
name	Especifica el nombre del módulo. Debe ser unix
path	Especifica la ruta del módulo objeto. Es indispensable
options	Especifica la lista de opciones soportadas por el modulo unix. No existen opciones obligatorias para este módulo.

La sección del módulo MYSQL es la sección utilizada para gestionar el módulo de cuentas de usuario de bases de datos mysql y está compuesta por las siguientes opciones:

Opción	Descripción
name	Especifica el nombre del módulo. Debe ser mysql
path	Especifica la ruta del módulo objeto. Es indispensable
options	Especifica la lista de opciones soportadas por el módulo mysql. Existe una opción obligatoria en este módulo llamada “dsn” que contiene la cadena de conexión a la base de datos mysql.

EJEMPLO DE ARCHIVO /etc/sump.conf

```
{
  "hostname":"localhost",
  "port":"1803",
  "user":"nobody",
  "group":"nobody",
  "log":"stdout",
  "clients":{
     "localhost":
"144a84776dae3697e477bbbf7103a4315b8609daa63cd98f3c4ad1ae7e67ce09"
   },
   "modules":[
                {"name":"unix",
                 "path":"/root/sumpd/libsump_unix.so",
                 "options":{"timeout":"300"}
                },
                {"name":"mysql",
                 "path":"/root/sumpd/libsump_mysql.so",
                 "options":{
                         "dsn":"user@pass/mysql"}
                }
        ]
}
```

Anexo 12 – PRUEBAS DEL PROTOCOLO SUMP

Descripción

Este documento contiene las instrucciones para realizar pruebas al protocolo sump

Requerimientos Mínimos

- ✓ Máquinas virtuales de simulación creadas y configuradas
- ✓ Servidor sump instalado
- ✓ Cliente sump instalado

Paso 01 – Crear Túneles SSH y Configurar Seguridad

Ejecutar los siguientes comandos con usuario root en el servidor openvz

```
cd /root/ sump/
./asegurar_servidor.sh
./crear_tunel_ssh.sh
```

Nota: El script crear_tunel_ssh.sh creara un canal seguro hacia todos los servidores sump. El script asegurar_servidor.sh revisa si el servidor ssh tiene soporte para el nivel de encriptación mayor a 1024 bits y aplica reglas firewall.

Nota: Las reglas de firewall del archivo asegurar_servidor.sh corresponden a un entorno virtualizado bajo containers, por lo que no serán las mismas en caso se aplique en un entorno de producción real. Para ese caso, los filtros deberán aplicarse por cada servidor con el software sump instalado y otorgar accesos solo a los servidores autorizados.

Paso 02 – Iniciar el servidor sump en las máquinas virtuales

Ejecutar los siguientes comandos con usuario root en cada máquina virtual

```
cd /root/sumpd
./sump
```

Paso 03 – Iniciar comunicación con el cliente sump

Ejecutar el siguiente comando en el servidor openvz.

```
cd /root/ sump
./cliente --url “sump://127.0.0.1:77XY/module/action/headers”
--pretty
```

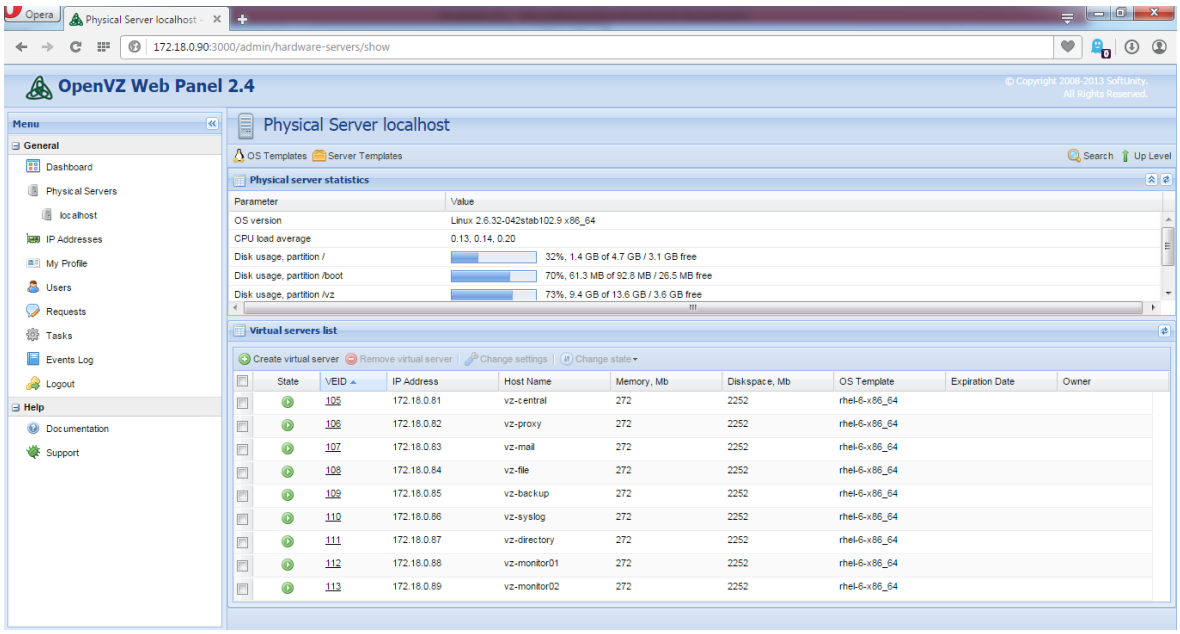
Donde XY es último octeto del número IP de cualquier máquina virtual que ejecuta el agente sump, <module/action/headers> corresponde a los comandos soportados por el servidor sump y que se pueden ver en la tabla 33.

Anexo 13 – EVIDENCIAS

Descripción

Este documento contiene las evidencias del funcionamiento del protocolo sump en el entorno virtualizado descrito en la Arquitectura de la Simulación descrito en el anexo 09.

01 – Infraestructura Virtual Utilizada



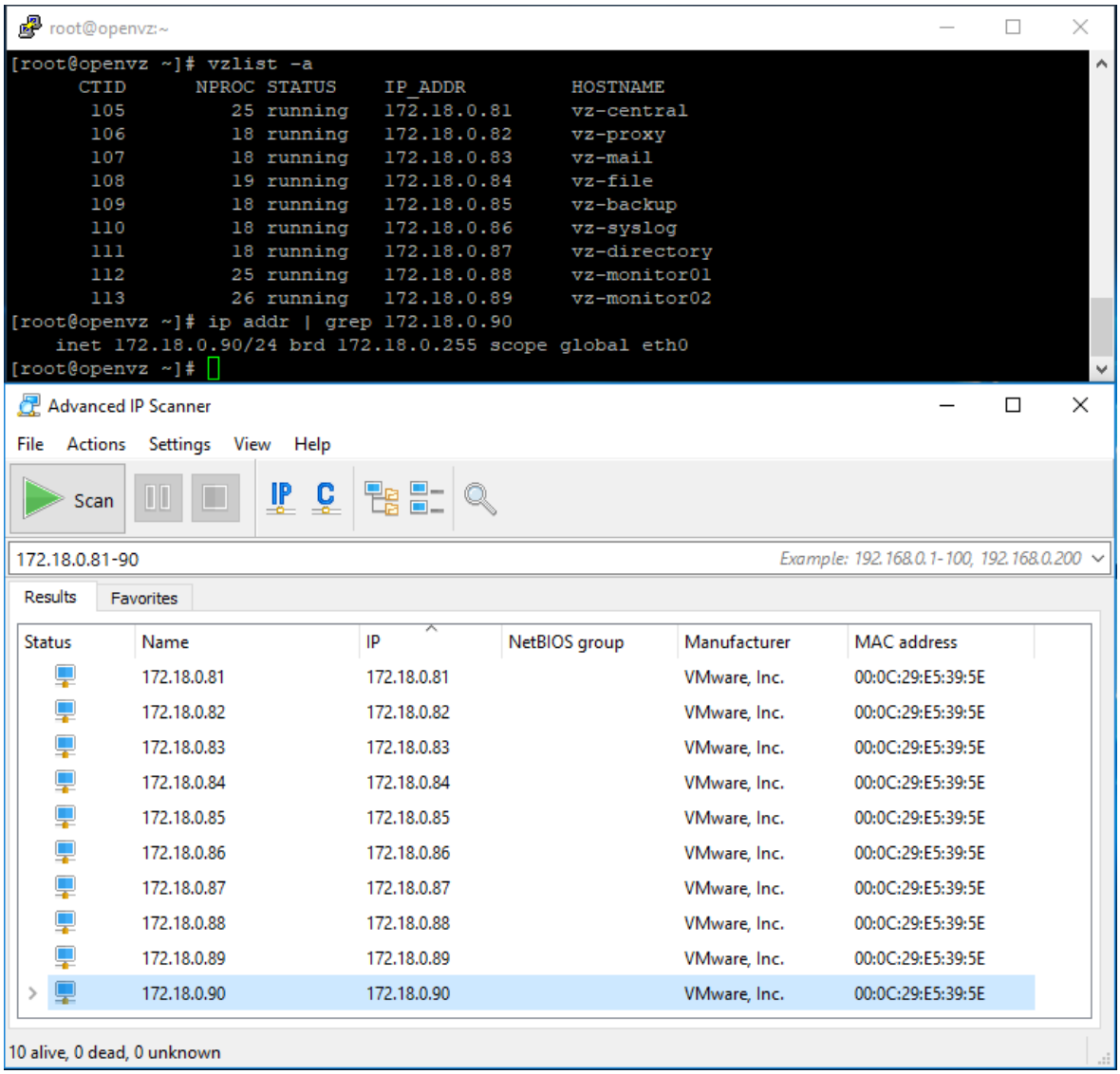
Detalles

En esta evidencia se pueden observar el buen funcionamiento de todos los elementos descritos en el anexo 05 - “Infraestructura Virtual”, que se detallan a continuación

- ✓ Servidor openvz con ip 172.18.0.90 (dirección ingresada en navegador)
- ✓ 09 containers con plantillas rhel-6-x86_64, en el segmento 172.18.0.0/24
- ✓ Estado activo de todos los containers

Nota: Para esta evidencia se utilizó el software Opera Browser.

02 – Mapeo de Red



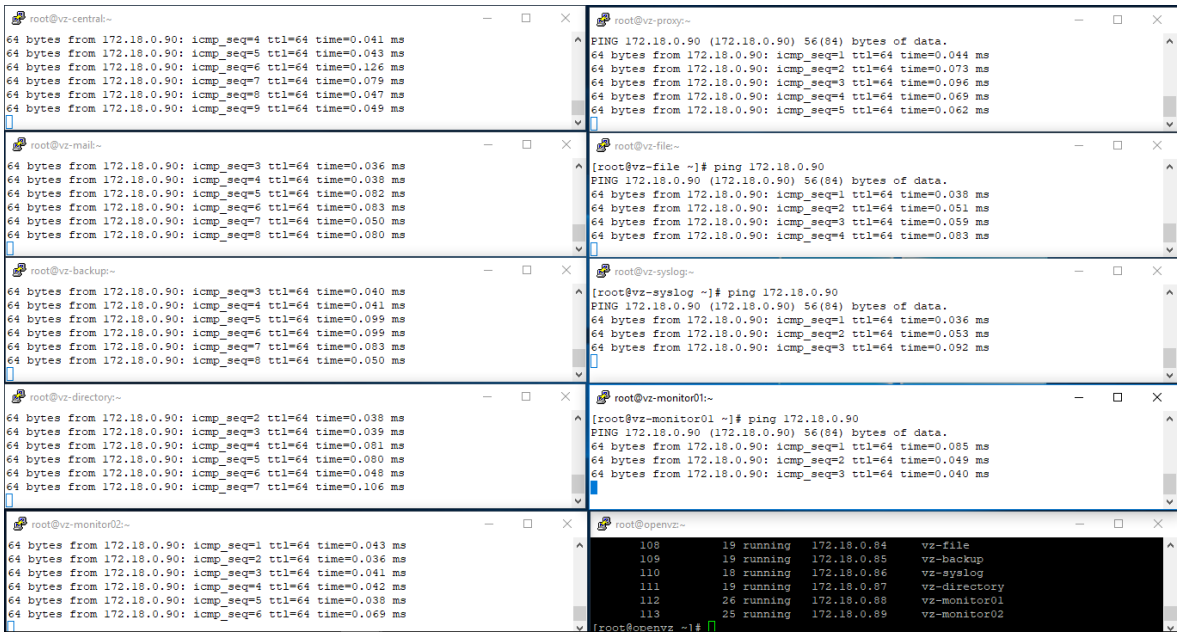
Detalles

En esta evidencia se pueden observar todos los equipos detectados en la red de simulación (segmento 172.18.0.0/24). Entre los elementos se destacan:

- ✓ Tecnología VMware
- ✓ Números IP en conformidad con el diagrama de Arquitectura de la Simulación del anexo 05

Nota: Para esta evidencia se utilizó el software Advance IP Scanner.

03 – Prueba de conectividad

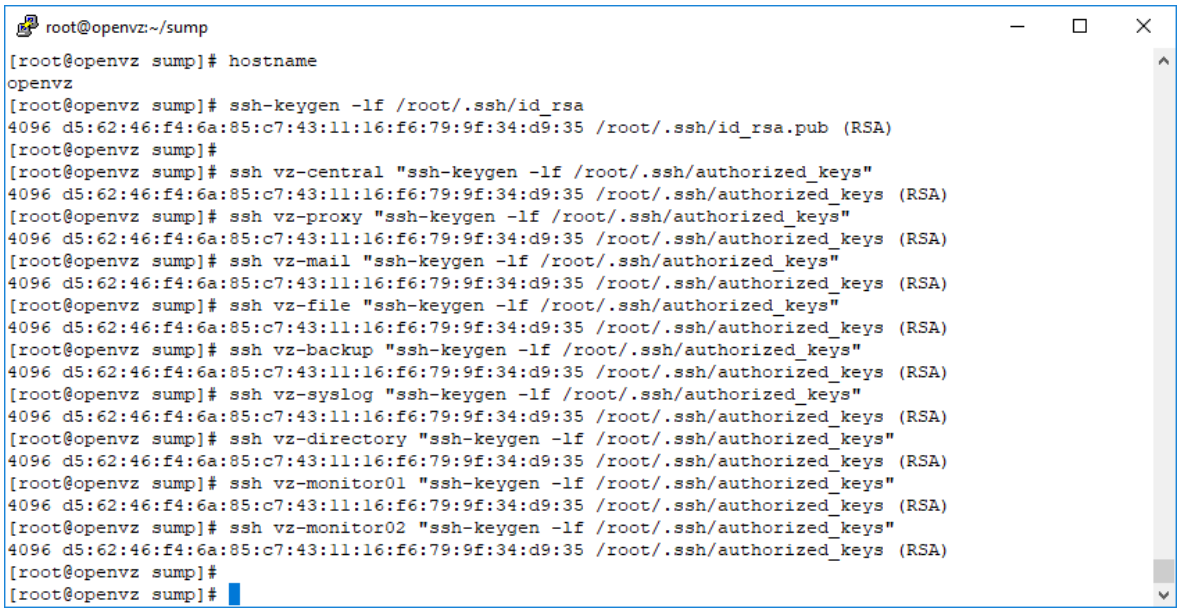


Detalles

En esta evidencia se pueden observar que todos los servidores utilizados en la simulación (ventanas blancas) pueden conectar mediante ping al servidor openvz cuyo ip es 172.18.0.90 (ventana negra).

Nota: Para esta evidencia se utilizó el software putty.

04 – Configuración de conexión segura



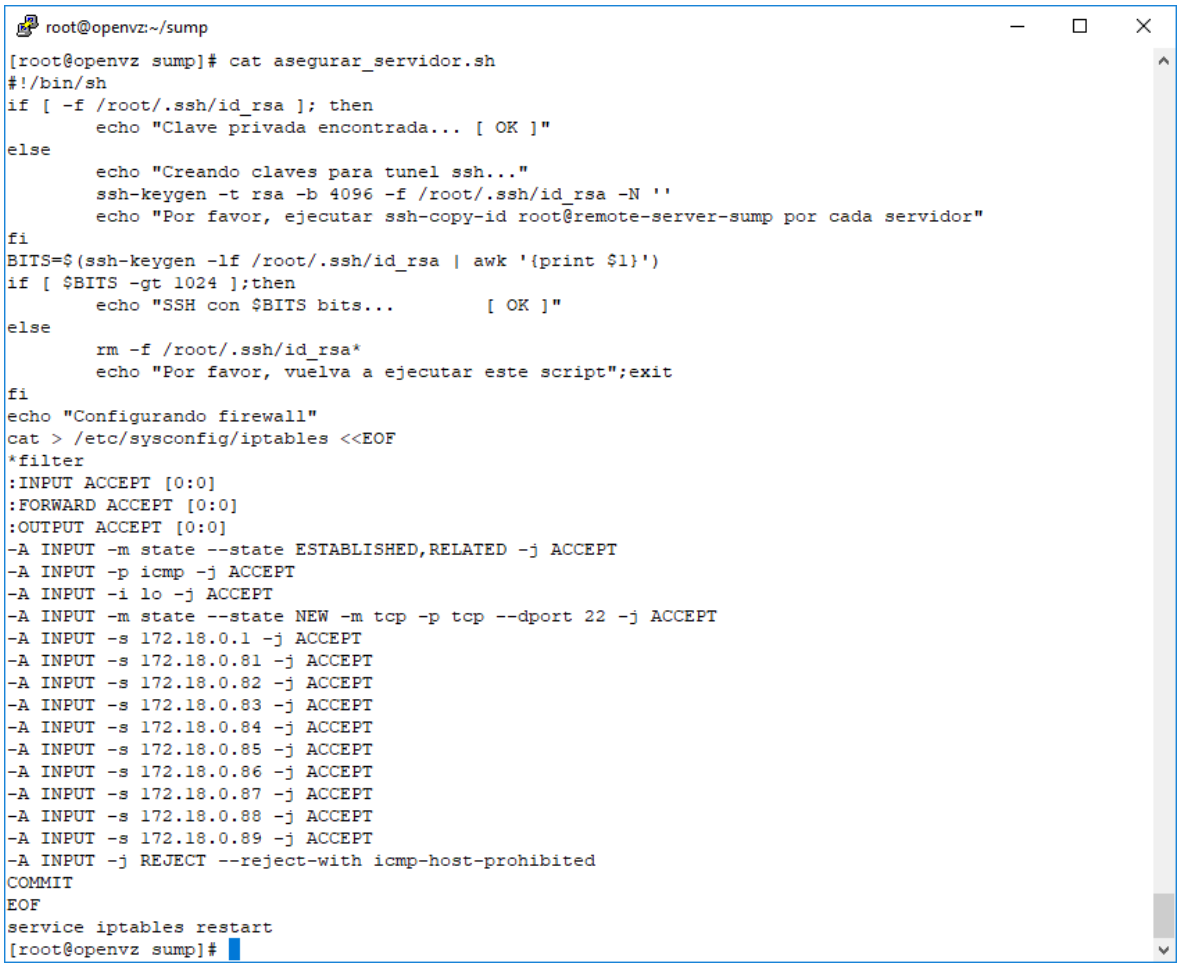
Detalles

En esta evidencia se pueden observar la configuración de las llaves de seguridad del sistema SSH entre el servidor openvz y los containers. Se puede observar:

- ✓ El algoritmo de encriptación es RSA para todos los servidores
- ✓ El nivel de encriptación es de 4096 bits para todos los servidores

Nota: Para esta evidencia se utilizó el software putty.

05 – Configuración de firewall



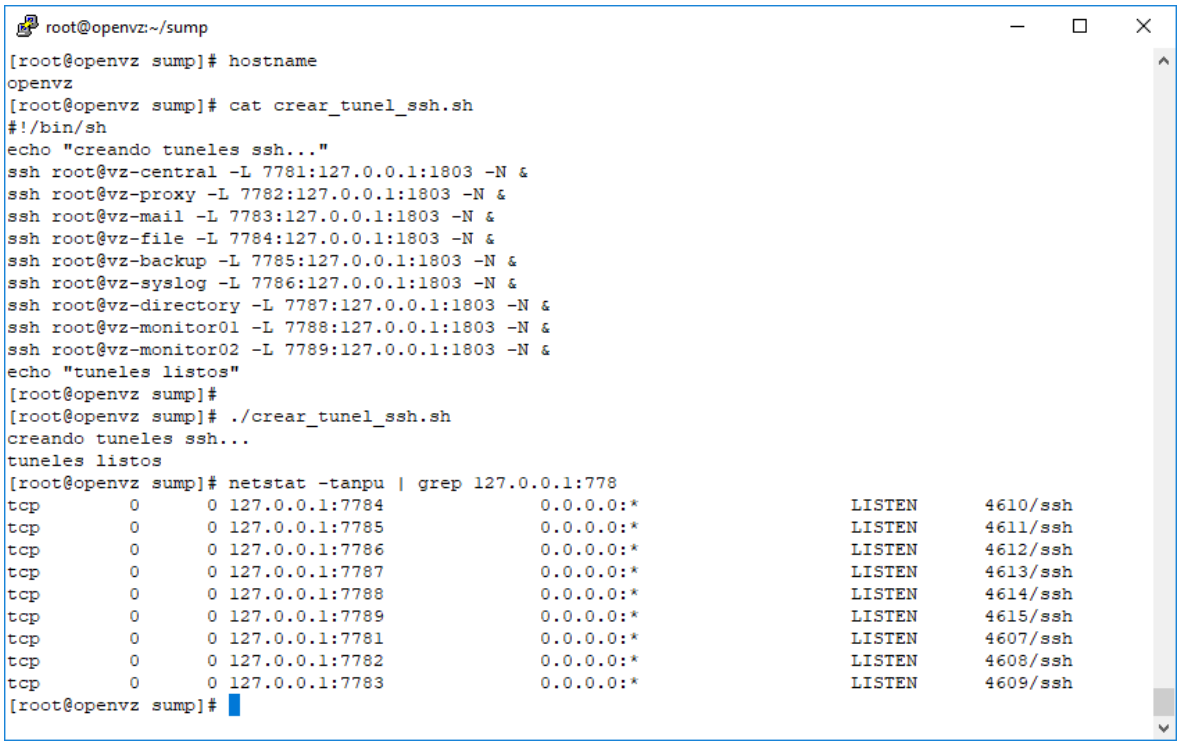
Detalles

En esta evidencia se pueden observar el script asegurar_servidor.sh utilizado para proteger las conexiones. Se puede apreciar:

- ✓ La validación del tamaño de clave mayor a 1024 bits
- ✓ Las reglas iptables para el servidor openvz

Nota: Para esta evidencia se utilizó el software putty.

06 – Túnel SSH



Detalles

En esta evidencia se pueden observar el script crear_tunel_ssh.sh utilizado para establecer túneles seguros entre el servidor openvz y los containers. Se puede apreciar lo siguiente:

- ✓ La inclusión de un túnel por cada container
- ✓ El uso del puerto 1803 como puerto de acceso al servicio SUMP
- ✓ Las conexiones establecidas por los túneles
- ✓ El protocolo SSH que soporta a los túneles

Nota: Para esta evidencia se utilizó el software putty.

07 – Cliente SUMP 1

Detalles

En esta evidencia se puede observar el funcionamiento del protocolo SUMP mediante las peticiones del software cliente. Se puede apreciar lo siguiente:

- ✓ El cliente es utilizado desde el servidor openvz
- ✓ El acceso al servidor vz-central mediante el puerto 7781 del túnel SSH
- ✓ El uso del módulo UNIX y MYSQL
- ✓ El uso del método GET_USER
- ✓ El resultado de las consultas en formato json identado

Nota: Para esta evidencia se utilizó el software putty.

08 – Cliente SUMP 2

Detalles

En esta evidencia se puede observar el funcionamiento del protocolo SUMP mediante las peticiones del software cliente en distintos containers. Se puede apreciar lo siguiente:

- ✓ El cliente es utilizado desde el servidor openvz
- ✓ El acceso al servidor vz-proxy mediante el puerto 7782 del túnel SSH
- ✓ El acceso al servidor vz-mail mediante el puerto 7783 del túnel SSH
- ✓ El uso del módulo UNIX
- ✓ El uso del método GET_USER
- ✓ El resultado de las consultas en formato json identado
- ✓ El resultado de una consulta errónea (501)

Nota: Para esta evidencia se utilizó el software putty.

09 – Cliente SUMP 3

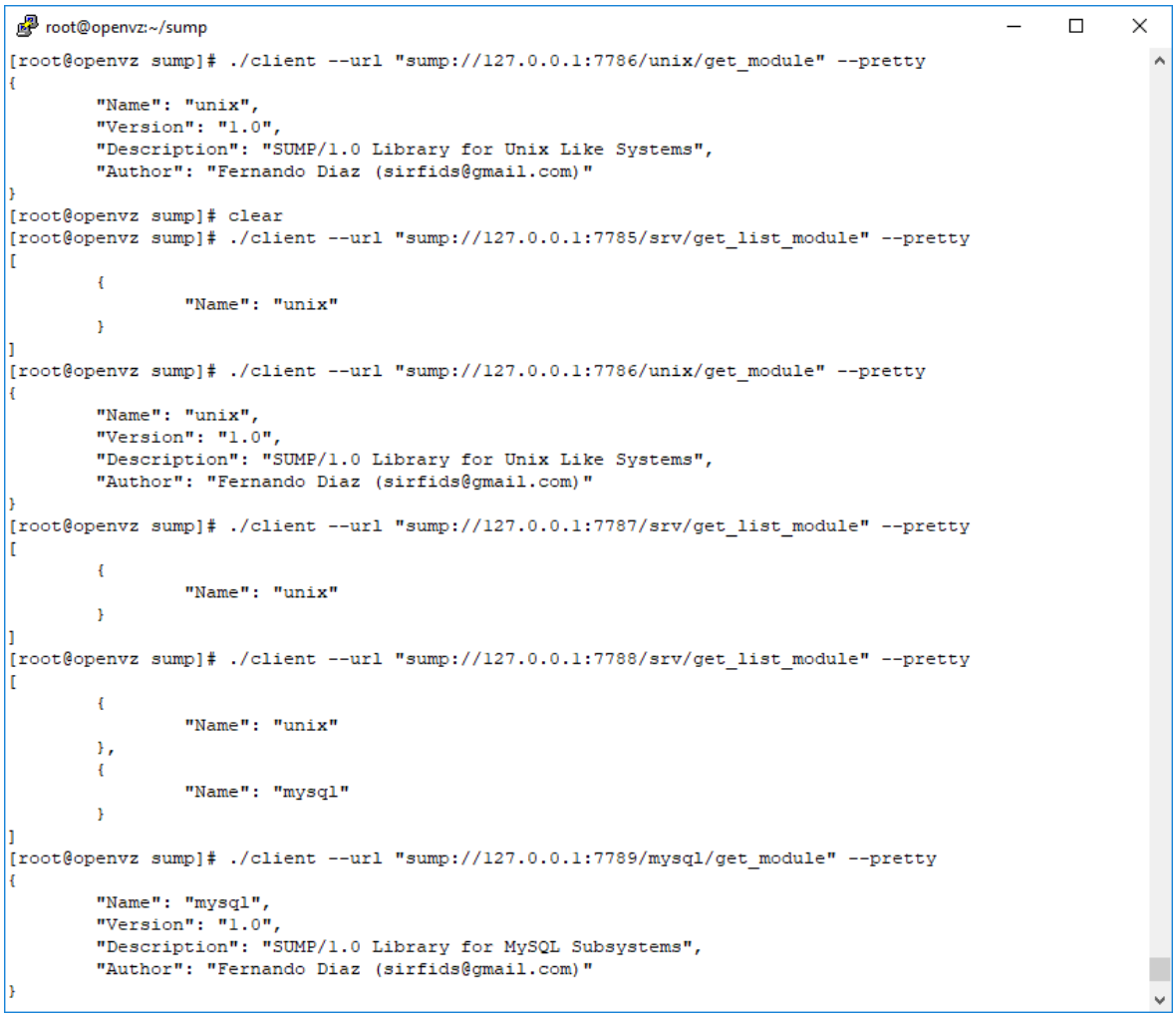
Detalles

En esta evidencia se puede observar el funcionamiento del protocolo SUMP mediante las peticiones del software cliente en distintos containers. Se puede apreciar lo siguiente:

- ✓ El cliente es utilizado desde el servidor openvz
- ✓ El acceso al servidor vz-file mediante el puerto 7784 del túnel SSH
- ✓ El acceso al servidor vz-backup mediante el puerto 7785 del túnel SSH
- ✓ El uso del módulo UNIX
- ✓ El uso del método GET_GROUP y GET_LIST_POLICY
- ✓ El resultado de las consultas en formato json identado

Nota: Para esta evidencia se utilizó el software putty.

10 – Cliente SUMP 4



Detalles

En esta evidencia se puede observar el funcionamiento del protocolo SUMP mediante las peticiones del software cliente en distintos containers. Se puede apreciar lo siguiente:

- ✓ El cliente es utilizado desde el servidor openvz
- ✓ El acceso al servidor vz-syslog mediante el puerto 7786 del túnel SSH
- ✓ El acceso al servidor vz-directory mediante el puerto 7787 del túnel SSH
- ✓ El acceso al servidor vz-monitor01 mediante el puerto 7788 del túnel SSH
- ✓ El acceso al servidor vz-monitor02 mediante el puerto 7789 del túnel SSH
- ✓ El uso del módulo UNIX, MYSQL y SRV
- ✓ El uso del método GET_MODULE, GET_LIST_MODULE
- ✓ El resultado de las consultas en formato json identado

Nota: Para esta evidencia se utilizó el software putty.

Anexo 14 – VALIDACION DE INSTRUMENTOS

Descripción

Este anexo contiene la validación de instrumentos por juicio de expertos y también los informes de recolección de datos realizados en este trabajo de investigación.

Documentos Proporcionados

- ✓ Documento resumen de validación de expertos
- ✓ Documento de validación de instrumentos por juicio de expertos
- ✓ Documento de requerimiento de información de sistemas y servicios de Petroperú sede Iquitos (OPS)
- ✓ Documento de requerimiento de información de priorización de sistemas y servicios de Petroperú sede Iquitos (OPS)
- ✓ Documento de requerimiento de información de servidores en la EMPRESA PETROPERÚ sede Iquitos (OPS)
- ✓ Documento de requerimiento de validación de información de Pre-Test
- ✓ Documento de requerimiento de validación de información de Post-Test

VALIDACION DE EXPERTOS

Descripción

El presente documento presenta un cuadro resumen de la validación de instrumentos por juicio de expertos.

Cuadro de Validación de Expertos

	DEFICIENTE		ACEPTABLE		BUENO		EXCELENTE	
	F	%	F	%	F	%	F	%
Congruencia de Ítems	0	0	0	0	2	66.66666667	1	33.33333333
Amplitud de contenido	0	0	0	0	0	0	3	100
Redacción de los Ítems	0	0	0	0	0	0	3	100
Claridad y precisión	0	0	0	0	1	33.33333333	2	66.66666667
Pertinencia	0	0	0	0	0	0	3	100
TOTAL	**0**	**0**	**0**	**0**	**3**	**20.00000000**	**12**	**80.00000000**

Fuente: Elaboración del investigador

Dónde:

F = Frecuencia

% = Porcentaje de frecuencia

Se puede observar que, de acuerdo a la validación de los expertos, los instrumentos utilizados durante el proyecto de investigación tienen la calificación total de **BUENO** en un 20% y una calificación total de **EXCELENTE** en un 80%.

Anexo 15 – PROPUESTA RFC

Descripción

Este documento contiene la especificación del protocolo SUMP siguiendo la guía de referencia para la elaboración según el RFC 7322.

Requisitos Mínimos del Formato RFC

- ✓ Debe estar redactado enteramente en inglés.
- ✓ Debe elaborarse en texto plano.

Debido al formato particular que tienen los documentos RFC, el contenido de este anexo empieza en la siguiente hoja.

Independent Submission F. Diaz Sanchez
Internet-Draft UCV
Intended status: Experimental March 26, 2018
Expires: September 27, 2018

A Simple User Management Protocol (SUMP)
rfc-sump

Abstract

The Simple User Management Protocol (SUMP) is an stateless application-level protocol for managing local users and privileges in different subsystems. This document covers an overview of its architecture and functioning as well as its terminology and syntax.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 27, 2018.

Table of Contents

1. Introduction

The SUMP protocol is an stateless application layer protocol whose purpose is to be light, secure and modular in order to manage users and privileges of any type of subsystems.

This protocol was born as an initiative to be able to manage in a faster and more secure way the access control in the different Petroperú's servers of its Selva Headquarters following the ISO/IEC 27002:2013 recommendations in their directives (9.2.5 and 13.1.2).

User management on systems is as old as the systems themselves. Over time, management solutions such as LDAP were created to enable centralized control of users, groups and privileges. But there are still numerous scenarios where it is common and even convenient to have local user accounts, for example users in Databases, Embedded Systems, Web Applications, etc.

Managing local users often has an impact on organizational security because it's easy to give unwanted access by mistake if controls aren't adequate.

The SUMP Protocol attempts to solve this particular problem by centralizing user information and local privileges into a quick and easy service for the technical staff in charge of this management, so that the visibility of these accounts and the privileges they have been granted is not lost.

2. Notational Conventions and Generic Grammar

This notational conventions and generic grammars are taken mostly from the HTTP protocol found in [RFC1945] and [RFC7230].

2.1. Augmented BNF

This document uses Augmented Backus-Naur Form (ABNF) described in [RFC5234] for Syntax Specifications, including all list extensions for compact definition of lists (see section 1.2 of [RFC7230]).

2.2. Basic Rules

The basic parsing construct used in this specification are described in RFC 1945 (see section 2.2 of [RFC1945])

Single-character quoting using the backslash ("\") character is not permitted in SUMP/1.0.

3. Protocol Specification

3.1. Protocol Parameters

3.1.1. SUMP Version

SUMP uses the same HTTP numbering scheme to indicate versions of the protocol (see section 3.1 of [RFC1945]).

```
SUMP-Version   = "SUMP" "/" 1*DIGIT "." 1*DIGIT
```

Proxy and gateway applications are not permitted.

3.1.2. Uniform Resource Identifiers

SUMP uses the same HTTP URI schemes to identify a network resource (see section 3.2 of [RFC1945])

3.1.2.1. sump URL

The "sump" scheme is the following

```
sump_URL      = "sump:" "//" host [ ":" port ] ( request )
host          = <A legal Internet host domain name
                 as defined by section 2.1 of [@!RFC1123] >
port          = *DIGIT
request       = ( "/" module "/" action [ "/" headers ] )
module        = ( mod_name | "srv" | ext-mod )
mod_name      = token
action        = ( user_action | group_action | mod_action
                    | policy_action )

user_action   = ( "get_user" | "get_list_user" | ext-ua )
group_action  = ( "get_group" | "get_list_group" | ext-ga )
module_action = ( "get_module" | "get_list_module" | ext-ma )
policy_action = ( "get_policy" | "get_list_policy" | ext-pa )

ext-mod       = token
ext-ua        = (token | "new_user" | "del_user" | "upd_user")
ext-ga        = (token | "new_group" | "del_group" | "upd_group")
ext-ma        = token
ext-pa        = (token | "new_policy" | "upd_policy")

headers       = param
param         = par-value [ *("&" par-value) ]
par-value     = variable "=" value
variable      = token
value         = 1*(ALPHA | DIGIT)
```

3.1.3. Character Sets & Character Encoding

SUMP uses the same definition of the term "character set" as that described for HTTP/1.0 (see section 3.4 of [RFC1945]).

The default charset used in SUMP is UTF-8.

3.1.4. Content Codings

The Content Coding are mostly used to compress payload information. To do this, the SUMP protocol follows the same rules applied in the HTTP/1.0 protocol (see section 3.5 of [RFC1945]). Its form is as follows:

```
content-coding   = token
```

This section describes the concept of Content Codings operations. However, the header that implements this concept is described in section 3.7.7 of this document where the behavior of "Content-Encoding" is covered.

There are 4 types of token: gzip, compress, deflate and identity (see section 3.5 of [RFC1945]).

3.1.5. Media Types

SUMP uses the same Media Types rules as those proposed by the HTTP/1.0 protocol (see section 3.6 of [RFC1945]).

There are some examples of media types preferred with SUMP/1.0:

```
text/html
text/json
text/xml
```

The section 3.7.6 covers the "Content-Type" header that uses this definition of media types.

3.2. SUMP Message

3.2.1. Message Types

SUMP messages consist of requests from client to server and responses from server to client.

```
SUMP-message   = Request | Response     ; SUMP/1.0 messages
```

There are 2 types of messages: Request and Response. Request (section 3.3) and Response (section 3.4) follow the message format described in section 4.1 of [RFC1945].

```
gen-sump-message = start-sump-line
                   *(sump-message-header CRLF)
                   CRLF
                   [ sump-message-body ]
start-sump-line  = Sump-Request-Line | Sump-Status-Line
```

3.2.2. Message Headers

SUMP header fields follow the same format as the proposed on HTTP/1.0 (see section 4.2 of [RFC1945]).

Some examples of headers fields format are:

```
Host: sump.myserver.com
Content-Encoding: gzip
Content-Type: text/json; charset=utf-8
```

The section 3.7 define all headers that SUMP/1.0 support.

3.2.3. Message Body

The sump-message-body is the information that the server delivers to the client. It may contain the data that the customer expects or it may be the description of an error that occurred during the process. The sump-message-body is only present in the Request.

```
sump-message-body = sump-entity-body
                  | <sump-entity-body encoded by Content-Encoding>
```

As a general rule, the sump-message-body can be summarized as follows:

```
sump-message-body := Content-Encoding(
                       Content-Type( sump-entity-body )
                     )
```

As can be seen, the sump-entity-body corresponds to what is known as Payload.

3.2.4. Message Length

The Message Length is only used when it is a Response and is calculated based on the length of the sump-entity-body. That is, when the Content-Encoding header is used, the value must be the encoded length to be sent.

The Content-Length header MUST be used always to indicate the this Message Length.

3.3. Request

All client messages are performed through the Request message, which consists of 2 well defined parts: a Sump-Request-Line followed by several sump-request-headers. These messages follow the same structure as the one defined by the HTTP/1.0 protocol (see section 5 of [RFC1945]).

```
Request  = Sump-Request-Line            ; Section 3.3.1
           *(sump-request-header CRLF)  ; Section 3.3.2
           CRLF
```

3.3.1. Sump-Request-Line

The Sump-Request-Line notation allows the client to define the method that wants to use directly and categorically. This application takes the following form.

```
Sump-Request-Line   = Method SP SUMP-Version CRLF
```

3.3.1.1. Method

It is the main part of the Sump-Request-Line and specifies the action to be executed on the server. SUMP/1.0 supports the following methods.

```
Method         = "GET_USER"                ; Section 3.5.1
               | "GET_LIST_USER"           ; Section 3.5.2
               | "GET_GROUP"               ; Section 3.5.3
               | "GET_LIST_GROUP"          ; Section 3.5.4
               | "GET_MODULE"              ; Section 3.5.5
               | "GET_LIST_MODULE"         ; Section 3.5.6
               | "GET_POLICY"              ; Section 3.5.7
               | "GET_LIST_POLICY"         ; Section 3.5.8
               | "SET_INFO"                ; Section 3.5.9
               | ext-sump-method
ext-sump-method = token
```

3.3.2. Sump-Request-Header Fields

The sump-request-header fields are elements of the Request that provide a precise context that will help the server to correctly interpret the request. The client MUST be able to translate the URL into the respective sump-request-header.

```
sump-request-header = Host                ; Section 3.7.1
                    | Key                 ; Section 3.7.2
                    | Module              ; Section 3.7.3
                    | User                ; Section 3.7.4
                    | Group               ; Section 3.7.5
                    | Policy              ; Section 3.7.6
                    | Policy-Type         ; Section 3.7.7
                    | Policy-Data         ; Section 3.7.8
                    | Format              ; Section 3.7.9
                    | Info                ; Section 3.7.10
                    | extension-request-header
extension-request-header = token
```

3.4. Response

The counterpart of the Request is the Response and is sent by the server in response to the client. Unlike the Request, the Response message is composed of a Sump-Status-Line, some sump-response-headers and a sump-entity-body as follows.

```
Response = Sump-Status-Line               ; Section 3.4.1
           *( sump-response-header CRLF)  ; Section 3.4.2
           CRLF
           sump-entity-body               ; Section 3.4.3
```

3.4.1. Sump-Status-Line

The Sump-Status-Line allows the customer to immediately identify the result of their Request. Its syntax is identical to the Status-Line of the HTTP/1.0 protocol (see section 6.1 of [RFC1945]).

```
Sump-Status-Line = SUMP-Version SP Sump-Status-Code CRLF
```

3.4.1.1. Sump-Status-Code

The Sump-Status-Code as well as the Status-Code of the HTTP/1.0 protocol (see section 6.1.1 of [RFC1945] is a 3-digit integer code that allows to categorize the server result.

The choice of the Sump-Status-Code as a mechanism to categorize the result issued by the server is due to its familiarity with the HTTP/1.0 protocol. However, it is important to mention that not all codes have an identical interpretation to their HTTP par, but rather close in context. Likewise, the Sump-Status-Code is a subset of all the codes that support the HTTP protocol.

The categories in Sump-Status-Code are the following:

- 2XX: Success
- 4XX: Client Error
- 5XX: Server Error

The individual codes for each category can be reviewed in section 3.6 and are presented below:

```
Sump-Status-Code  =
      "200"  ; Section 3.6.1: OK
    | "204"  ; Section 3.6.1: No Content
    | "400"  ; Section 3.6.2: Bad Request
    | "401"  ; Section 3.6.2: Unauthorized
    | "403"  ; Section 3.6.2: Forbidden
    | "404"  ; Section 3.6.2: Not Found
    | "405"  ; Section 3.6.2: Method Not Allow
    | "406"  ; Section 3.6.2: Format Not Supported
    | "413"  ; Section 3.6.2: Request Too Long
    | "429"  ; Section 3.6.2: Too Many Request
    | "431"  ; Section 3.6.2: Header Too Large
    | "432"  ; Section 3.6.2: Bad Header
    | "500"  ; Section 3.6.3: Internal Error
    | "501"  ; Section 3.6.3: Not Implemented
    | "503"  ; Section 3.6.3: Unavailable
    | "504"  ; Section 3.6.3: Timeout
    | "505"  ; Section 3.6.3: Version Not Supported
    | ext-sump-code

ext-sump-code = 3DIGIT
```

3.4.2. Sump-Response-Header Fields

The sump-response-header allows the server to provide relevant information about the transmission of the entity-body to the client. The client MUST be able to interpret and apply the necessary modifications to the entity-body in order to capture the payload.

```
sump-response-header = Content-Type       ; Section 3.7.11
                     | Content-Enconding   ; Section 3.7.12
                     | Content-Length      ; Section 3.7.13
                     | ext-sump-response-header
ext-sump-response-header = token
```

3.4.3. Entity Body (Payload)

The sump-entity-body is the message containing the information that the server has collected in response to the client's request. This message does not necessarily contain the information that is expected (for example, it may contain a 403 response), however, the SUMP protocol ensures that there is a response.

```
sump-entity-body    = *OCTET
```

A sump-entity-body MUST be included with the Response message.

3.5. Method Definitions

There are four types of information that is supported by SUMP:

- User: Represents a user account with or without any kind of privileges or attributes in order to use a subsystem

- Group: Represents a logical group of Users

- Module: Represents a subsystem supported by the protocol

- Policy: Represents a set of privileges for Users or Groups

3.5.1. GET_USER

The method GET_USER retrieve information about a User of a particular Module.

3.5.2. GET_LIST_USER

The method GET_LIST_USER retrieve information about a list of Users of a particular Module.

The format of an entity-body for GET_LIST_USER and GET_USER is the same. In Fact, GET_USER is a 'list of users' with just one account.

3.5.3. GET_GROUP

The method GET_GROUP retrieve information about a Group of a particular Module.

3.5.4. GET_LIST_GROUP

The method GET_LIST_GROUP retrieve information about a list of Groups of a particular Module.

The format of an entity-body for GET_LIST_GROUP and GET_GROUP is the same. In Fact, GET_GROUP is a 'list of groups' with just one group.

3.5.5. GET_MODULE

The method GET_MODULE retrieve relevant information about a Module, like name, version, authors, descriptions, etc.

3.5.6. GET_LIST_MODULE

The method GET_LIST_MODULE retrieve information about all modules supported by the server but in a shorter way, just the list of names.

This method is only possible with the special module name 'srv'.

3.5.7. GET_POLICY

The method GET POLICY retrieve information about a particular privileges of a particular User or Group of just one Module.

3.5.8. GET_LIST_POLICY

The method GET LIST POLICY retrieve information about all privileges rules supported by an specific Module.

3.5.9. SET_INFO

The SET_INFO method manipulates information about users, groups and policies. To keep the protocol simple, it has been considered to encapsulate these actions into a single method. However, the client SHOULD separate the following actions: new_user, del_user, upd_user, new_group, del_group, upd_group, new_policy and upd_policy and translate it to the method SET_INFO form.

3.6. Status Code Definitions

SUMP use the same status code of HTTP/1.0 but in a limited way in order to provide a very well-known form to identify errors and status of the request. It's important to understand that this status code not always has the same meaning as HTTP but related.

3.6.1. Successful 2XX

This sump-status-code indicates that the client's request was correctly received, translated and resolved.

200 OK

The request was attended without problems

204 No Content

The request was properly resolved but there are no contents to show.

3.6.2. Client Error 4XX

This sump-status-code indicates that the client's request had some kind of error and cannot be accepted or resolved.

404 Bad Request

The request contains syntactic or semantic errors.

401 Unauthorized

The request come from a device (client) not authorized, the Key was not sended or is different to the Key registered on the server.

For security reasons, the response has to be the same in both scenarios.

403 Forbidden

The request was valid, but the server refuses to process it due to an explicit prohibition in the server configuration or module.

404 Not Found

The request was valid, but the server could not find the requested information.

405 Method Not Allow

The method requested by the client is unknown to the server. The response MUST include (on the sump-entity-body) a list of methods that the server accepts.

406 Format Not Supported

The Content-Type requested by the client in the Format header is unknown to the server. The response MUST include (on the sump-entity-body) a list of supported formats.

413 Request Too Long

The size of the Request is larger than expected by the server and therefore cannot be processed.

429 Too Many Request

The 429 (Too Many Request) status code indicates that the server is refusing to process a request because the limit of connections

permitted in a particular period of time was reached. This is a native way to prevent abuse of the service.

431 Header Too Large

The 431 (Header Too Large) status code indicates that the server is refusing to process a Request because at least one sump-request-header is larger than expected by the server and therefore cannot be processed.

432 Bad Header

One or all of the sent headers by the client are not known to the server. The response MUST include (on the sump-entity-body) a list of supported headers.

3.6.3. Server Error 5XX

This sump-status-code indicates that the server is having trouble resolving the request.

500 Internal Server Error

The server found an unexpected event or condition that forced it to abandon the request resolution.

501 Not Implemented

The server does not support the functionality (module) required by the client.

503 Unavailable

The server is not in a position to respond to the request due to the current load, maintenance plan or unavailability of any related service on which it depends.

504 Timeout

The 504 (Timeout) status code indicates that the server cannot be able to resolve the request in the time proposed on its configuration.

505 Version Not Supported

The server does not support or reject the version specified in the Sump-Request-Line

507 Disk Full

The 507 (Disk Full) status code indicates that the server is currently unable to handle the request due to a disk full condition.

3.7. Header Field Definitions

3.7.1. Host

The "Host" header field is the same as the one defined in the HTTP/1.1 protocol (see section 5.4 of [RFC7230]).

```
Host = sump-uri-host [ ":" port ]
sump-uri-host = <host, see section 3.2.2 of [@!RFC3986]>
```

The client MUST to use the Host header field to establish the connection with the server.

3.7.2. Key

The "Key" header field in a request provides the secret key which identify the client from the server. It is recommended that the Key does not contain a real password, but rather an SHA256 Hash.

This Key and the client Host MUST be registered on the server. If the keys didn't match, the server MUST respond with a 403 (Forbidden) status code.

3.7.3. Module

The "Module" header field in a request provides the name of the subsystem in which the server has to seek the information. If the sump-request-header doesn't contains a User or Group header field and if the method is GET_USER or GET_GROUP or GET POLICY, the server MUST respond with a 400 (Bad Request) status code.

3.7.4. User

The "User" header field in a request provides the exact information about a particular user that the server has to seek. The client MUST build the Request content.

3.7.5. Group

The "Group" header field in a request provides the exact information about a particular group that the server has to seek. The client MUST build the Request content.

3.7.6. Policy

The "Policy" header field in a request provides the name of the privilege or permission in a subsystem that can be apply to a User or Group. This header cannot be included on GET_LIST_POLICY method.

3.7.7. Policy-Type

The "Policy-Type" header field in a request provides the type of objective for the Policy header. The values can be "User" or "Group". If another values are sent, the server MUST respond with a 432 (Bad Request) status code.

3.7.8. Policy-Data

The "Policy-Data" header field in a request provides information about the user or group whose permission or privilege indicated in the Policy header needs to be reviewed.

3.7.9. Format

The "Format" header allows the server to know which Content-Type the client wishes to receive.

Example of URL with format:

```
sump://sump.server.com/unix/get_user/user=fids/format=json;utf-8
```

Example of the Request Header with Format for this URL

```
GET_USER SUMP/1.0
Host: sump.server.com
Key: CF799C07995FBAB0B882992089AEA52E60927E8E19DBDCC7215EC819A0F6FA93
Module: unix
User: fids
Format: json;utf-8
```

The "Key" value in this example is just an aleatory value.

3.7.10. Info

The "Info" header allow the server to know the action to be executed and the data to be able to carry it out. The actions allowed are 'cmd=new', 'cmd=del' and 'cmd=upd'. The parameters to be able to execute these commands must be sent according to the "param" scheme. Each module of the server can include new parameters according to its convenience. The client MUST provide a mechanism so that sensitive

information such as user keys does not appear in plain text in the URL.

Example of URL with format:

sump://sump.server.com/unix/del_user/user=test

Example of the Request Header with Format for this URL

```
SET_INFO SUMP/1.0
Host: sump.server.com
Key: CF799C07995FBAB0B882992089AEA52E60927E8E19DBDCC7215EC819A0F6FA93
Module: unix
Info: cmd=del&user=test
```

The "Key" value in this example is just an aleatory value.

3.7.11. Content-Type

The Content-Type header field is the same as the one defined in the HTTP/1.0 protocol (see section 10.5 of [RFC1945]).

Content-Type = "Content-Type" ":" media-type

Media types are defined in Section 3.2.1. An example of the field is

Content-Type: text/json; charset="utf-8"

3.7.12. Content-Encoding

The Content-Encoding header field is the same as the one defined in the HTTP/1.0 protocol (see section 10.3 of [RFC1945]).

Content-Encoding = "Content-Encoding" ":" content-coding

Content codings are defined in Section 3.5. An example of its use is

Content-Encoding: gzip

3.7.13. Content-Length

The Content-Length header field is the same as the one defined in the HTTP/1.0 protocol (see section 10.4 of [RFC1945]).

Content-Length = 1*DIGIT

3.8. Representation of Data Exchange

SUMP communications are protocol text follow the rules described on sections 3.1 to 3.7. However, the internal representation of information MUST be contained in high-level data structures that make up the PDUs (Packet Data Units) for the Request and Response. The purpose of this design is to provide the flexibility to support future serialization of data using others notations such as ASN.1, protocol buffers, etc.

3.8.1. The Request-PDU

The "Request-PDU" is a data structure to store a Request. The client MUST translate this structure in the protocol data exchange mechanism (protocol text by default) and the server MUST process in the inverse order to store properly.

This PDU is inspired in the protocol data units from SNMP (see section 5 of [RFC1098]).

The C structure of this PDU is the follow:

```
struct varBind{               // Variable Binding
   char name[256];
   char* value;
};

struct varBindList{           // List of Variable Bindings
   struct varBind **info;     //  Info List
   unsigned short nelem;      //  Number of Elements
};

struct REQUEST{               // REQUEST PDU
   char method[33];           //  Method
   char protocol[12];         //  Protocol
   char version[6];           //  Version
   struct varBindList header; //  Headers
};
```

Figure 1

The implementor can choose another language if its more convenient.

3.8.2. The Response-PDU

The "Response-PDU" is a data structure to store a Response. The client MUST translate this structure in the protocol data exchange

mechanism (protocol text by default) and the server MUST process in the inverse order to store properly.

This PDU is inspired in the protocol data units from SNMP (see section 5 of [RFC1098]).

The C structure of this PDU is the follow:

```
struct varBind{               // Variable Binding
   char name[256];
   char* value;
};

struct varBindList{           // List of Variable Bindings
   struct varBind **info;     //  Info List
   unsigned short nelem;      //  Number of elements
};

struct sc{                    // Status Code
   char code[4];              //  Code
   char* msg;                 //  Message
};

struct RESPONSE{              // RESPONSE PDU
   char method[33];           //  Method
   char protocol[12];         //  Protocol
   char version[6];           //  Version
   struct sc status;          //  Status of Response
   struct varBindList header; //  Headers
   char* body;                //  Payload
};
```

Figure 2

The implementor can choose another language if its more convenient.

4. Security Considerations

SUMP/1.0 purpose is deal with critical information like users and privileges. This is one of the most wanted information who every hacker seek for. Therefore, the developer must take the necessary measures to protect it. Among the general security policies that must be complied with are the following.

- The encryption of the communications can be develop inside of the client/server or outside by specialized software. The data exchange MUST be included. SSL and another old technologies are forbidden.

- Local information in the server, cannot be stored on filesystem, but in memory instead. Users, Groups and Privileges are the most valuable local information to protect.

- The server has to be capable to identify, at least, suspicious activities like too many request. The server MAY take actions against the client like disconnections or temporary blocking.

5. Acknowledgements

This document reuse many of the HTTP/1.0 and HTTP/1.1 specifications defined by HTTP authors. This is because SUMP was designed based mostly on HTTP protocol. The author is grateful for the availability of these documents and acknowledges the efforts made by their creators, who in turn join in thanking the other contributors to this protocol.

Likewise, the author acknowledges the contribution of the SNMP protocol in the design of the SUMP protocol, and reiterates his admiration and appreciation for all those involved on that project.

6. Normative References

[RFC1098] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1098, DOI 10.17487/RFC1098, April 1989, <https://www.rfc-editor.org/info/rfc1098>.

[RFC1945] Berners-Lee, T., Fielding, R., and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, DOI 10.17487/RFC1945, May 1996, <https://www.rfc-editor.org/info/rfc1945>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

Author's Address

Fernando Díaz Sánchez
UCV
Calle 8 de Octubre 254
Chiclayo - Peru

Phone: +51 979-350-599
Email: sirfids@gmail.com