DISEÑAR E IMPLEMENTAR UN SISTEMA INTEGRADO DE PUNTOS DE ACCESO WI-FI.

ANTONIO MARÍA PÉREZ

Trabajo de profundización para optar por el título de Magister en Ingeniería Electrónica

DIRECTOR:

Ing. Luis Carlos Trujillo Arboleda, M.SC

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

MAESTRÍA EN INGENIERÍA ELECTRÓNICA

BOGOTA

NOVIEMBRE DE 2017

Tabla de contenido

1. INTRODUCCIÓN

Los puntos de acceso *Wi-Fi* permiten la conectividad en cualquier momento y lugar donde se tenga cobertura [1]. Su sencillez y facilidad de uso hacen que rápidamente se estén desplegando en lugares que tienden a atraer a los usuarios nómadas. A pesar de que los puntos de acceso tienen un alcance limitado, un costo menor de instalación y sus anchos de banda de operación son más altos que otras alternativas como las redes inalámbricas 3G/4G, su uso en muchos puntos tiende a una baja utilización y no son rentables[2]. Esta baja utilización no se debe a la incompatibilidad tecnológica si no a otros factores, que inducen a un lento despliegue de nuevos puntos. Algunos factores que influyen son:

- La falta de rentabilidad para la sostenibilidad del servicio.
- El incremento de herramientas que vulneran la seguridad de las redes Wi-Fi. Por ejemplo: Aircrack, AirSnort, Kismet, Cain & Able, WireShark, Fern Wi-Fi Wireless Cracker, CoWPAtty, Airjack, WepAttack, NetStumbler, inSSIDer, Wifiphisher, entre otros[3].
- La vulnerabilidad al compartir datos privados y públicos en el mismo segmento de red.
- La baja oferta en el mercado de productos Wi-Fi con portales cautivos embebidos.
- La dependencia de gestión humana para la creación de cuentas de usuario Wi-Fi.
- La ausencia de diversos métodos de autenticación para acceder al servicio Wi-Fi por medio de una interfaz web.
- La falta de oportunidad de poder obtener información de los usuarios que acceden al servicio Wi-Fi.
- La carencia de poder emitir publicidad controlada a través de un portal cautivo.
- La necesidad de invertir en muchos dispositivos para brindar un servicio Wi-Fi a través de un portal cautivo.

Con el propósito de resolver las causas antes descritas y poder aumentar el uso de puntos de acceso Wi-Fi para servicio público, es necesario diseñar e implementar en un único dispositivo, un sistema integrado de puntos de acceso Wi-Fi con gestión de portales cautivos que permita las siguientes bondades técnicas:

- Conexión a redes Wi-Fi abiertas.
- Soporte de diversos métodos de autenticación para acceder al servicio Wi-Fi por medio de portales cautivos:
    - ✓ Portal cautivo con capacidad de despliegue de un formulario personalizable con el objeto de poder capturar datos del usuario.
    - ✓ Portal cautivo con capacidad de acreditar usuarios detrás de un sistema de autenticación digital descentralizado (OpenID)[4]. Servicio que permite identificarse ante una variedad de sitios web, que poseen las credenciales de un usuario y que son de uso reiterativo para él.

    - ✓ Portal cautivo usuario/contraseña. Esquema que permite la conexión a bases de datos internas o externas donde se encuentran las credenciales del usuario, con posibilidad de recuperación de contraseñas a través de correo electrónico.
    - ✓ Portal cautivo con acceso directo. Opción con la capacidad de conectar un determinado número de usuarios simultáneos sin necesidad de tener que escribir algún tipo de credencial. Muy funcional para escenarios que solo quieren desplegar información sin necesidad de capturar algún tipo de dato del usuario.
- Despliegue de publicidad o información corporativa controlada no invasiva.
- Seguridad a través de un servidor AAA (autenticación, autorización y contabilización) interno o externo, bajo el estándar IEEE 802.1X[5].
- No requiere de un controlador central o máquinas virtuales para el despliegue del servicio de un portal cautivo.
- Capacidad de desplegar hasta dos portales cautivos simultáneos. Necesario cuando las credenciales de los usuarios vienen de dos fuentes diferentes de autenticación. Por ejemplo: Bases de datos locales y OpenID.
- Sistema operativo abierto. Permitirá la continuación y evolución en el desarrollo del producto.

Algunos fabricantes reconocidos de la industria como *Cisco Systems, Hewlett-Packard, Aruba Networks* y *Ruckus*, poseen soluciones mixtas de Access Point con Wireless LAN Controller (dispositivo que se encarga de controlar todos los Access Point adscritos a él), que permiten algunas de las características técnicas anteriormente descritas y con la posibilidad de desplegar portales cautivos para requerimientos corporativos de gran escala. Estas soluciones no son funcionales para pequeños negocios o puntos de acceso con un área de cobertura limitada. No obstante, existe en la industria Access Point de tipo autónomo (dispositivo no dependiente de un tercero), que permiten ejecutar el servicio de un portal cautivo con autenticación de credenciales de manera local. Algunos de estos dispositivos son:

- *LINKSYS LAPAC1750PRO* Business Access Point Wireless: Permite establecer un portal cautivo personalizable, sin embargo, solo posibilita ciertos métodos de autenticación predefinidos. La personalización está limitada a texto e imágenes dentro de la misma página de gestión. La configuración se debe hacer por acceso directo al equipo o mediante una controladora de la misma marca. No dispone de un API (interfaz de programación de aplicaciones) para agregar más desarrollos propios a la funcionalidad del equipo.
- Aruba Networks IAP 105: Línea de Access Point autónomo, llamados IAP (punto de acceso instantáneo), con posibilidad de configurar un portal cautivo cuyo propósito es solamente la autenticación del usuario a través de credenciales soportadas sobre una base de datos local dentro del dispositivo.

- La compañía *Cradlepoint* y *NetComm Wireless*: Son Access Point autónomos con un portal cautivo embebido con el beneficio del 3G/4G, pero con sistema operacional cerrado. Solo permiten procesos de OEM para su comercialización.

Para el desarrollo de este trabajo se planteó como objetivo general diseñar e implementar un sistema integrado, flexible y seguro para portales cautivos Wi-Fi. Dentro del cual se desarrollaron los siguientes objetivos específicos:

- Diseñar un sistema de portal Wi-Fi que integre cuatro métodos de acceso con despliegue de publicidad controlada no invasiva, un servidor AAA y una interfaz de gestión web para toda la solución.
- Implementar un prototipo del sistema de portal Wi-Fi.
- Validar la funcionalidad y el desempeño del prototipo implementado.
- Validar las credenciales de un usuario a través de los diferentes métodos de portales cautivos, confirmando su acceso o denegación a la red restringida.

2. MARCO TEÓRICO

2.1. PORTAL CAUTIVO

Un portal cautivo[6] es una página web que el usuario de una red de acceso público está obligado a ver e interactuar antes de que se le conceda el acceso. Los portales cautivos suelen ser utilizados por los centros de negocios, aeropuertos, hoteles, cafeterías y otros lugares que ofrecen puntos de acceso Wi-Fi gratis para los usuarios de Internet.

Cuando un usuario potencial inicia sesión en una red con un portal cautivo, se encuentra una página Web que requiere ciertas acciones antes de que se conceda acceso a Internet. Un portal cautivo simple obliga al usuario a mirar al menos (si no lee) una página de política de uso y luego hacer clic en un botón que indica que está de acuerdo con los términos de la política. Presumiblemente, esto puede ayudar a absolver al proveedor de responsabilidad en caso de que el usuario cometa una actividad criminal u otra actividad destructiva mientras está conectado. En algunos portales cautivos, se muestran anuncios para los patrocinadores del proveedor y el usuario debe hacer clic en ellos o cerrar las ventanas en las que aparecen antes de acceder a Internet. Algunos portales cautivos requieren la entrada de un ID de usuario y una contraseña previamente asignados antes de acceder a Internet. Tal autenticación puede desalentar el uso de puntos calientes inalámbricos como sitios para llevar a cabo actividades criminales. La mayoría de los servidores con portales cautivos incluyen programas antivirus y cortafuegos para ayudar a proteger las computadoras de los usuarios de Internet y entre sí.

Algunas personas pueden conectarse repetidamente, utilizando la red de forma casi continua para descargar música, videos u otros archivos grandes. Esta actividad, llamada acaparamiento de ancho de banda, puede minimizarse mediante programación adicional en el portal cautivo. Dicha programación puede controlar la velocidad de descarga de los archivos grandes, limitar el tamaño (en kilobytes o megabytes) de los archivos que se pueden descargar, restringir el número de descargas que pueden ocurrir en una sola sesión o bloquear la conexión a los sitios web utilizados comúnmente para descargar archivos grandes.

2.2. RADIUS (*Remote Authentication Dial-In User Service)*

RADIUS[7] es un protocolo cliente-servidor que permite que los servidores remotos se comuniquen con un servidor central para autenticar y autorizar los usuarios al sistema o servicios solicitados. RADIUS permite a una empresa mantener perfiles de usuario en una base de datos centralizada que todos los servidores remotos pueden compartir. Tener un servicio centralizado permite que sea más fácil hacer un seguimiento del uso para la facturación y para mantener las estadísticas de la red. RADIUS fue creado por Livingston y estandarizado por la IETF *(Internet Engineering Task Force*) en la RFC 2865.

Un servidor de acceso a red (NAS - Network Access Server) funciona como un cliente de RADIUS[8]. El cliente es responsable de pasar la información de usuario a los servidores RADIUS, y luego actuar sobre la respuesta que se devuelve.

Los servidores RADIUS son responsables de recibir las solicitudes de la conexión del usuario, autenticar al usuario y luego devolver información de configuración necesaria para que el NAS se la envíe al usuario.

Las transacciones entre el NAS y el servidor RADIUS son autenticadas mediante el uso de una llave compartida, que nunca es enviada a través de la red. Además, cualquier contraseña de usuario es enviada encriptada entre el NAS y el servidor RADIUS.

El servidor RADIUS puede soportar varios métodos para autenticar un usuario. Cuando se proporciona el nombre de usuario y contraseña, puede soportar *PPP (Point-to-Point Protocol)*, *PAP* (*Password Authentication Protocolo), CHAP (Challenge Handshake Authentication Protocol)*, inicio de sesión UNIX y otros mecanismos de autenticación.

La comunicación entre el NAS y el servidor RADIUS usa *User Datagram Protocol* (UDP)[9].

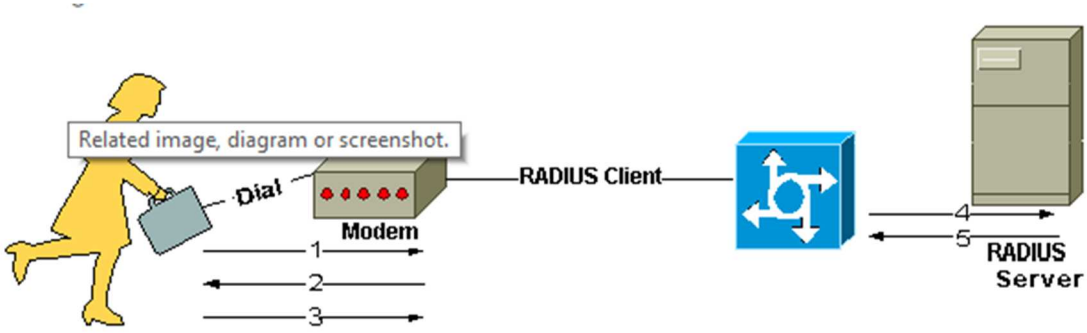

Fig. 1. Interacción entre usuario, cliente y servidor RADIUS. Tomada de [9]

En Fig**. 1** se muestra el proceso de interacción entre usuario, NAS (cliente RADIUS) y servidor NAS:

1. El usuario inicia autenticación al NAS.
2. NAS solicita el nombre de usuario y la contraseña (si el protocolo de autenticación de contraseña [PAP]) o reto (challenge) (si es el protocolo de autenticación de desafío [CHAP]).
3. Usuario responde.
4. El cliente RADIUS (NAS) envía nombre de usuario y contraseña cifrada al servidor RADIUS.
5. El servidor RADIUS responde con Aceptar, Rechazar o Desafiar.
6. El cliente RADIUS actúa sobre los servicios y los parámetros de servicios incluidos con Aceptar o Rechazar.

El servidor RADIUS puede ser un servidor AAA (Authentication, Authorization and Accountig).

2.3. AAA (Authentication, Authorization and Accountig)

AAA[10] es un término para referirse a tres funciones de seguridad: autenticación, autorización y contabilización:

- Autenticación (*authentication*). Proporciona el método de identificación de usuarios. Es la forma en que un usuario se identifica antes de poder acceder a la red y los servicios que esta ofrece. Usualmente el proceso de autenticación se realiza con usuario y contraseña.
- Autorización (*authorization*). Determina si el usuario está permitido para realizar o acceder a ciertas tareas, recursos o servicios.
- Contabilización (*accounting*). Mide los recursos que un usuario consume durante su acceso, por ejemplo, la cantidad de tiempo o datos que el usuario ha enviado o recibido durante su sesión.

2.4. WiFi: REDES INALAMBRICAS 802.11

Las redes inalámbricas[11] son cada vez más populares, y los hogares, oficinas, cafés, bibliotecas, aeropuertos, zoológicos, y otros lugares públicos están siendo dotados con ellas para conectar computadores, tabletas y teléfonos inteligentes a internet. Las redes inalámbricas también se pueden usar para permitir que dos o más computadoras se comuniquen sin usar internet. El estándar principal para redes inalámbricas es 802.11, también conocido como WiFi.

Las redes 802.11 se pueden usar en dos modos. El modo más popular es conectar clientes, como computadoras portátiles y teléfonos inteligentes, a otra red, como una intranet de la empresa o Internet. Este modo se muestra en **Fig. 2(a)**. En el modo de infraestructura, cada cliente está asociado con un AP (punto de acceso) que a su vez está conectado a la otra red. El cliente envía y recibe sus paquetes a través del AP. Varios puntos de acceso pueden conectarse entre sí, generalmente mediante una red cableada llamada sistema de distribución, para formar una red 802.11 extendida. En este caso, los clientes pueden enviar tramas a otros clientes a través de sus AP.

El otro modo, que se muestra en **Fig. 2** (**b**), es una red ad hoc. Este modo es una colección de computadoras que están asociadas para que puedan enviarse tramas directamente entre sí, y no hay puntos de acceso.

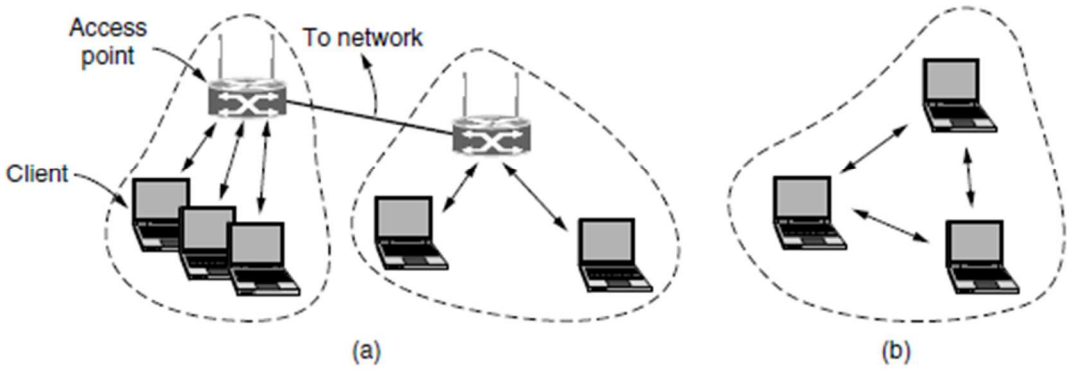

Fig. 2. Arquitectura 802.11. (a) Modo infraestructura. (b) Modo ad hoc. Tomada de [11]

Todos los protocolos 802, incluidos 802.11 y Ethernet, tienen una cierta estructura en común. Una vista parcial de la pila del protocolo 802.11 se muestra en **Fig. 3**. La pila es la misma para clientes y AP. La capa física corresponde bastante bien con la capa física OSI, pero la capa de enlace de datos en todos los protocolos 802 se divide en dos o más subcapas. En 802.11, la subcapa MAC (*Medium Access Control*) determina cómo se asigna el canal, es decir, quién puede transmitir a continuación. Arriba está la subcapa LLC (*Logical Link Control*), cuyo trabajo es ocultar las diferencias entre las diferentes variantes 802 y hacerlas indistinguibles en lo que respecta a la capa de red. Esto podría haber sido una gran responsabilidad, pero en la actualidad el LLC es una capa de que identifica el protocolo (por ejemplo, IP) que se lleva dentro de una trama 802.11.

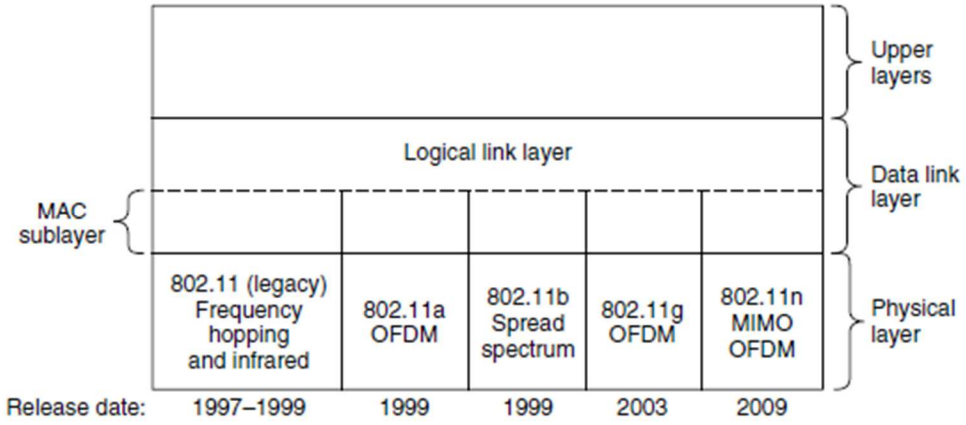

Fig. 3. Vista parcial de la pila del protocolo 802.11. Tomada de [11]

Se han agregado varias técnicas de transmisión a la capa física ya que 802.11 ha evolucionado desde que apareció por primera vez en 1997. Dos de las técnicas iniciales, el infrarrojo como los controles remotos de televisión y el salto de frecuencia en la banda de 2,4 GHz, ahora han desaparecido. La tercera técnica inicial, el espectro ensanchado de secuencia directa a 1 o 2 Mbps en la banda de 2,4 GHz, se amplió para funcionar a velocidades de hasta 11 Mbps y rápidamente se convirtió en un éxito. Ahora se conoce como 802.11b.

Nuevas técnicas de transmisión basadas en el esquema OFDM (*Orthogonal Frequency Division Multiplexing*) se introdujeron en 1999 y 2003. El primero se llama 802.11a y utiliza una banda de frecuencia diferente, 5 GHz. El segundo se quedó con 2.4 GHz y compatibilidad. Se llama 802.11g. Ambos dan tasas de hasta 54 Mbps.

En el año 2009, se adicionaron técnicas de transmisión que usan simultáneamente múltiples antenas en el transmisor y en el receptor. Con cuatro antenas y canales más amplios, el estándar 802.11n define velocidades de hasta 600Mbps

3. ESPECIFICACIONES

El proyecto realizado consiste en diseñar e implementar un prototipo de Access Point, que permite reducir en un solo dispositivo, la mayor cantidad de infraestructura posible, tanto en hardware como en software, en la implementación de servicios Wi-Fi. Este dispositivo permite proveer diferentes maneras de poder acceder a una red restringida por medio de portales cautivos que usan un servidor AAA para los servicios de autenticación como se muestra en **Fig. 4**. El usuario envía su usuario y contraseña al portal cautivo, este portal envía la petición a RADIUS, quien realiza la autenticación, autorización y contabilización mediante un servidor AAA y si es un portal cautivo con proveedor externo, se utiliza un servidor *LDAP* o directorio activo.

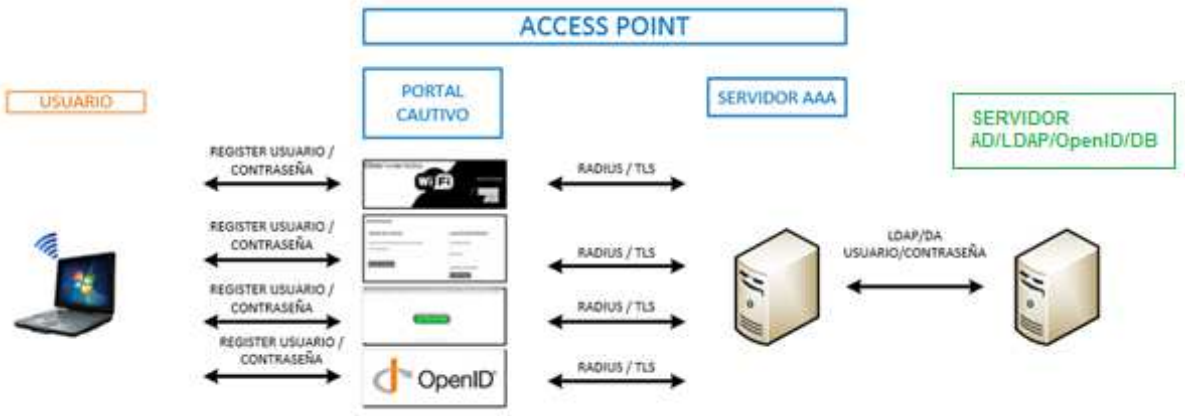

Fig. 4. Proceso de validación de credenciales

La integración y la seguridad ayudan a incrementar el despliegue de nuevos puntos de acceso Wi-Fi, principalmente en pequeños negocios que requieran sistemas sencillos y confiables que le permitan captar información de usuarios, distribuir contenidos, generar relaciones a largo plazo y aumentar así el grado de satisfacción.

El Access Point puede configurar y personalizar dos instancias, permitiendo una autenticación versátil desde diferentes fuentes donde se alojen las credenciales de los usuarios, para cualquiera de los siguientes portales cautivos:

- Generación de formularios para captura de datos.
- Acreditación de usuarios detrás de un sistema de autenticación digital descentralizado (OpenID).
- Conexión a través de una base de datos local por medio de usuario y contraseña.
- Acceso directo sin captura de credenciales.

El dispositivo es autónomo, a menos que el proceso de autenticación de usuarios requerido se ejecute desde un servidor AAA externo. Tiene los suficientes recursos (memoria, capacidad de procesamiento y almacenamiento) para soportar la integración de las siguientes características técnicas:

- Sistema operativo abierto.
- Servidor AAA (FreeRADIUS).
- Portales cautivos (CoovaChilli).
- Aplicaciones complementarias de software libre (HostAPD, SSH, Haserl, Mysql, Postfix, CherryPy y Samba).
- Personalización de los portales cautivos por medio de hojas de estilo en cascada (CSS).
- Despliegue de publicidad o información corporativa controlada no invasiva.
- Direccionamiento del usuario a un sitio web personalizado después de ser validado.
- Posibilidad de desconectar usuarios autenticados.
- Interfaz de gestión web para todo el sistema.

Se verificará mediante un protocolo de pruebas el funcionamiento y desempeño de dos diferentes métodos de portales cautivos ejecutados al mismo tiempo en el Access Point, validando las credenciales de un usuario a través de los diferentes modos de configuración, confirmando su acceso o denegación a la red restringida.

3.1. ESPECIFICACIONES FUNCIONALES DE CADA MÓDULO

A continuación, se describirán las especificaciones funcionales de los portales cautivos con Proveedor Público, con acceso directo sin captura de credenciales para la autenticación, con conexión por medio de usuario y contraseña, y para captura de datos por medio de formularios.

3.1.1. ESPECIFICACIONES FUNCIONALES PARA EL PORTAL CAUTIVO CON PROVEEDOR PÚBLICO

Este portal cautivo permite la autenticación a través de la cuenta de *Facebook*, *twitter* o *google* de un usuario y debe cumplir con las especificaciones dadas en la **Tabla 1**.

Tabla 1. Especificaciones funcionales para el portal cautivo con proveedor público

Requerimiento	Características de desempeño
Establecer una conexión segura con el API publica de autenticación del proveedor externo	Recibir un token de sesión del proveedor externo
Contrastar las credenciales del usuario que se está autenticando con las credenciales del API del proveedor externo	Recibir un ID del usuario identificado desde el proveedor externo
Asociar el ID del usuario con un grupo de privilegios en el sistema RADIUS	Recuperar un esquema de atributos asociado al usuario según su grupo

Generar la respuesta de autenticación correcta en el sistema RADIUS (ACCEPT). Nota: Existen tres tipos de posibles respuestas: ACCEPT/REJECT/CHALLENGE	Crear la respuesta de autenticación correcta cargada con los atributos asociados y enviarla al proceso de autorización de RADIUS Nota: tres (3) procesos de RADIUS (AAA) : Autenticación / Autorización / Contabilización
Superar el proceso de autorización por defecto	Enviar la respuesta ACCEPT al NAS solicitante
Autenticar y autorizar al usuario en el sistema cautivo	Permitir al usuario superar el portal cautivo y acceder al servicio de navegación
Registrar datos de usuario	EL NAS envía datos de la sesión del usuario que son registrados por el segmento Post-Auth de RADIUS

3.1.2. ESPECIFICACIONES FUNCIONALES PARA EL PORTAL CAUTIVO CON PROVEEDOR EXTERNO

Este portal cautivo permite la autenticación a través de las credenciales de usuario almacenadas en un servidor *LDAP* (Microsoft *AD*, *OpenLDAP*, *Oracle LDAP*, etc), cumpliendo con las especificaciones dadas en la **Tabla 2**.

Tabla 2. Especificaciones funcionales para el portal cautivo con proveedor externo

Requerimiento	Características de desempeño
Establecer una conexión segura con el servidor de autenticación LDAP	Recibir un Test-Data exitoso de conexión con el servidor LDAP
Contrastar el nombre de usuario con el referente local de autenticación	Definir que el usuario existe y que pertenece a un proveedor externo tipo LDAP
Recuperar los privilegios del grupo RADIUS asociado	Crear la respuesta de autenticación correcta cargada con los atributos asociados según el grupo
Cifrar la contraseña de usuario según el esquema del proveedor LDAP	Superar el segmento de Autenticación y pasar el par (Username, Password) al segmento de Autorización
Contrastar las credenciales del usuario con el proveedor externo	Recibir el token de validación de credenciales exitoso
Superar el proceso de autorización	Enviar la respuesta ACCEPT al NAS solicitante
Autenticar y autorizar al usuario en el sistema cautivo	Permitir al usuario superar el portal cautivo y acceder al servicio de navegación
Registrar datos de usuario	EL NAS envía datos de la sesión del usuario que son registrados por el segmento Post-Auth de RADIUS

3.1.3. ESPECIFICACIONES FUNCIONALES PARA EL PORTAL CAUTIVO CON ACCESO DIRECTO SIN CAPTURA DE CREDENCIALES DE AUTENTICACIÓN

En este portal cautivo, la autenticación de un usuario se realiza con base en credenciales predefinidas en la base de datos y debe cumplir con las especificaciones dadas en la **Tabla 3**.

En este modelo existe una variante donde el sistema RADIUS reconoce las credenciales del usuario según la MAC address del suplicante en lugar de un par (usuario, contraseña) por defecto; para llevar a cabo esto basta con definir un par (usuario [MAC], contraseña [MAC]) asociado al módulo RADA (RADIUS Authenticated Device Access); ya que el HOST RADIUS evalúa prioritariamente la autenticación por dispositivos.

Tabla 3. Especificaciones funcionales para el portal cautivo con acceso directo

Requerimiento	Características de desempeño
Definir un grupo de atributos RADIUS para un ***único usuario*** que se usara como credencial por defecto	El sistema de credenciales local cuenta con un grupo de atributos asociado a un único usuario para ser usado como credencial por defecto
Enviar credenciales vacías al proceso de autenticación del sistema RADIUS	RADIUS detecta las credenciales vacías y recupera el par (Username, Password) según el grupo de atributos definido para las credenciales por defecto
Superar el proceso de autorización	Enviar la respuesta ACCEPT al NAS solicitante
Autenticar y autorizar al usuario en el sistema cautivo	Permitir al usuario superar el portal cautivo y acceder al servicio de navegación
Registrar datos de usuario	EL NAS envía datos de la sesión del usuario que son registrados por el segmento Post-Auth de RADIUS

3.1.4. ESPECIFICACIONES FUNCIONALES PARA EL PORTAL CAUTIVO CON CONEXIÓN POR MEDIO DE CREDENCIALES ALMACENADAS

Este portal cautivo permite la autenticación a través de un usuario y contraseña almacenados en una base de datos SQL y debe cumplir con las especificaciones dadas en la **Tabla 4.**

En este modelo existe una variante donde se usa una base de datos remota que no está diseñada específicamente para ser compatible con el sistema RADIUS, en tal caso basta con definir una conexión ODBC con la base de datos foránea y que en ella exista como mínimo una vista (consulta que se presenta como una tabla [virtual]) que permita recuperar las credenciales del usuario.

Cuando se trabaja con una base de datos foránea, no se realizan consultas de autenticación directamente sobre esta, en su lugar se usa el esquema de vistas disponible para sincronizar las credenciales de usuario, en una base de datos específica para RADIUS; esto se debe a que las bases de datos foráneas pertenecen a sistemas de información críticos y por tanto no pueden ser modificadas estructuralmente para soportar todas las transacciones que requiere el sistema RADIUS.

Tabla 4. Especificaciones funcionales para el portal cautivo con acceso por RDBMS

Requerimiento	Características de desempeño
Definir el esquema de conexión con el servidor RDBMS (Relational Database Management System)	Recibir un Test-Data exitoso de conexión con el servidor RDBMS
Enviar el par (Username, Password) cifrado bajo esquemas Radix2 (protocolo de capa de aplicación diseñado) al proceso de autenticación del sistema RADIUS	RADIUS detecta las credenciales en Radix2 y recupera el par (Username, Password) en base a una consulta SQL de la tabla Rad-User
Recuperar los privilegios del grupo RADIUS asociado	RADIUS recupera los atributos relacionados con el usuario mediante una consulta SQL sobre la tabla Rad-Group
Superar el proceso de autorización	RADIUS verifica la contraseña asociada al usuario mediante una consulta SQL sobre la tabla Rad-User-Check. Enviar la respuesta ACCEPT al NAS solicitante
Autenticar y autorizar al usuario en el sistema cautivo	Permitir al usuario superar el portal cautivo y acceder al servicio de navegación
Registrar datos de usuario	EL NAS envía datos de la sesión del usuario que son registrados por el segmento Post-Auth de RADIUS

3.1.5. ESPECIFICACIONES FUNCIONALES PARA EL PORTAL CAUTIVO PARA CAPTURA DE DATOS POR MEDIO DE FORMULARIOS

Este portal cautivo permite la captura de información básica del usuario, como nombre, correo electrónico y teléfono, para ser almacenada en una base de datos *SQL*. Para lo cual debe cumplir con las especificaciones dadas en la **Tabla 5.**

Es importante mencionar que el sistema RADIUS puede ser configurado para desplegar formularios externos, siempre y cuando estos sean accesibles desde el *GARDEN* (sitios que no requieren un proceso de autorización) del portal cautivo y retornen un par (Username, Password) como resultado de su correcto procesamiento.

Tabla 5. Especificaciones funcionales para el portal cautivo con captura de datos

Requerimiento	Características de desempeño
Definir un esquema de formularios y atributos en el APP local de gestión de registro de datos.	Formularios asociados a un grupo de atributos dentro del sistema RADIUS.
Desplegar el formulario de captura de datos en el portal cautivo.	El sistema RADIUS procesa y despliega una vista HTML que contiene el formulario de registro de datos activo para el portal cautivo.
Procesar el formulario.	El sistema RADIUS recibe los datos del formulario y los procesa mediante una APP predefinida.
Genera la credencial de autorización con base en los datos capturados.	El sistema RADIUS genera una credencial permanente o volátil con base en los datos capturados.
El portal cautivo envía una petición de acceso al sistema RADIUS con el par (username, password) generado al completar el proceso de captura de datos	El sistema RADIUS recibe una petición de acceso desde el portal cautivo y la procesa según el esquema descrito en la tabla 4.

4. DESARROLLO DEL PROYECTO

4.1. ELEMENTOS

4.1.1. DEFINICIÓN

En términos generales es posible definir un elemento como aquella propiedad o recurso que da respuesta a un requerimiento concreto; bajo esta perspectiva, es necesario definir "requerimiento" antes de adentrarse en la formalización de los elementos; así pues, es posible referirse a:

- Requerimientos funcionales: Identifican las prestaciones fundamentales del sistema, la forma en que este debe comportarse frente a escenarios concretos y su relación directa con las acciones y expectativas del usuario.
- Requerimientos no funcionales: Definen las propiedades emergentes del sistema, tales como la persistencia, eficiencia y fiabilidad de los procesos que este realiza; así como, sus características de cara a la experiencia de usuario, a saber: accesibilidad, usabilidad y tiempo de respuesta.

Con base en lo anterior es posible referirse tanto a elementos funcionales como no funcionales, así mismo, teniendo en cuenta las características de la solución, resulta prudente clasificar los elementos con base en su naturaleza, ya sea como componentes de hardware o de software.

4.1.2. ELEMENTOS FUNCIONALES

A continuación, se mencionan los elementos funcionales más relevantes para la solución, soportándose en su importancia y en el contexto más general posible.

4.1.2.1. Elementos funcionales de Hardware

- Placa base: Ofrece el soporte electrónico principal de la solución, proveyendo tanto las capacidades de procesamiento, como el almacenamiento volátil requerido para desplegar todos los componentes lógicos necesarios para llevar a cabo las labores objetivo del sistema.
- Tarjeta de red inalámbrica: Provee el soporte electrónico para difundir redes inalámbricas, cumpliendo con ello el aspecto más relevante de la solución.

4.1.2.2. Elementos funcionales de Software

- HostAPD: Ofrece el soporte lógico necesario para crear y gestionar redes de datos inalámbricas, logrando que la solución actué como un punto de acceso WiFi independiente.
- Portal Cautivo: Provee el componente lógico principal con que interactúa el usuario, permitiendo definir y controlar las diferentes modalidades de autenticación que ofrece el sistema; así como, proveyendo el control del flujo de datos que se traduce en el acceso final del usuario a los recursos de la red que se encuentra tras el punto de acceso.
- Servidor AAA: Provee los servicios esenciales de seguridad de la solución, integrándose con el portal cautivo para dar respuesta a las peticiones del usuario final; así mismo, constituye el motor general de procesos que se usa como núcleo para el despliegue y operación de todos los componentes adicionales, que se traducen en las características de valor agregado de la solución.

4.1.3. ELEMENTOS NO FUNCIONALES

A continuación, se mencionan los elementos no funcionales más relevantes para la solución, según su importancia y en el contexto más general posible.

4.1.3.1. Elementos no Funcionales de Hardware

- Antena: Define el alcance, potencia y calidad de la señal emitida por la solución, definiendo en términos generales la experiencia de usuario en lo referente tanto al acceso al software subyacente, como a los recursos de la red o la navegabilidad en internet.

- Tarjeta SSD: Provee el soporte físico de almacenamiento persistente, para todos los componentes de software, así mismo un medio de almacenamiento completo y necesario para desplegar las características adicionales del sistema; supliendo con ello las limitaciones del almacenamiento integrado en la placa base.
- Case: Constituye el revestimiento externo de la solución, que se encarga tanto de contener y proteger los componentes electrónicos internos, como de definir las prestaciones físicas del sistema, a saber: tolerancia a temperaturas, capacidades de anclaje y resistencia a impactos.

4.1.3.2. Elementos no Funcionales de Software

- Base de datos SQL: Provee el soporte lógico de almacenamiento persistente, permitiendo almacenar, recuperar y actualizar de forma sencilla y eficiente el conjunto global de datos con que debe trabajar la solución. Así mismo, constituye la base que posibilita al sistema para ofrecer servicios subyacentes tales como accesos de bajo nivel para auditoría, gestión de copias de seguridad o integración con sistemas externos que consuman datos vía conectores SQL.
- Workers: Representa el conjunto general de scripts, aplicaciones y servicios que se ejecutan en segundo plano, permitiendo la integración de los componentes mayores y por ende manteniendo al sistema general unido y operativo.
- Servidor SAMBA: Ofrece la integración con las soluciones de directorio activo de Microsoft. Constituyendo un pilar fundamental de la solución de cara a ofrecer soporte para múltiples proveedores de credenciales.
- Conector ODBC: Permite a la solución integrarse con diferentes motores de bases de datos bajo un esquema general independiente. Constituye un pilar fundamental ya que le permite al sistema no solo integrarse con servicios externos bajo un protocolo neutral, sino también la capacidad de realizar procesos de sincronización y cooperación en tiempo casi real con dichos servicios.
- Conector LDAP: Ofrece la integración con las soluciones de controladores de dominio que sigan el estándar POSIX. Constituyendo un pilar fundamental de la solución de cara a ofrecer soporte para múltiples proveedores de credenciales.
- WEB API: Permite a la solución cooperar con diferentes sistemas externos por medio de una interfaz unificada, genérica e independiente de la implementación interna del sistema. Así mismo, provee la capacidad de consumir servicios de autenticación externos como los ofrecidos por Google o Facebook.
- Webmin: Constituye el centro de control general y unificado del sistema, ofreciendo al usuario administrador la capacidad de configurar, auditar y gestionar la solución a través de una única herramienta.

4.2. PROTOCOLOS

El funcionamiento de la solución planteada se soporta sobre los siguientes protocolos:

- **RADIUS (UDP 1812 y 1813):** Permite la comunicación y operatividad entre la entidad AAA (Servidor RADIUS) y el NAS Virtual (Portal Cautivo).
- **HTTP y HTTPS (TCP 80 y 443):** Permite la navegación web de los usuarios, así como la comunicación de agentes externos con el API WEB.
- **Protocolo de acceso a la base de datos.** Depende del motor de base de datos puntual que se implemente; ofrece el servicio de almacenamiento de datos, tanto para la entidad AAA, así como para los servicios de consulta y gestión del API WEB.

4.3. DIAGRAMAS UML (UNIFIED MODELING LANGUAGE)

Es necesario considerar que la presente solución no es un producto de software concreto, sino un conjunto de muchos servicios y aplicaciones, de fuentes diversas que se han ajustado para cooperar coordinadamente de tal forma que puedan operar como un sistema estable, dando con ello respuesta a los objetivos planteados.

Teniendo en cuenta lo anterior y el hecho de que UML está concebido idealmente para representar piezas de software concreto, construidas idealmente bajo el enfoque de la programación orientada a objetos; es preciso aclarar que los diagramas que se presentan a continuación no buscan en ningún momento ser “fieles” o “exhaustivos” respecto a los lineamientos de UML, ya que la presente solución no es una pieza de software concreto y la mayoría de sus componentes principales distan mucho de tener un enfoque orientado a objetos.

Los siguientes diagramas buscan ilustrar de forma global la estructura del sistema planteado, usando para ello las herramientas de representación de UML, sin pretender en ningún caso, llegar a ser diagramas UML formales.

4.3.1. DIAGRAMA DE CLASES

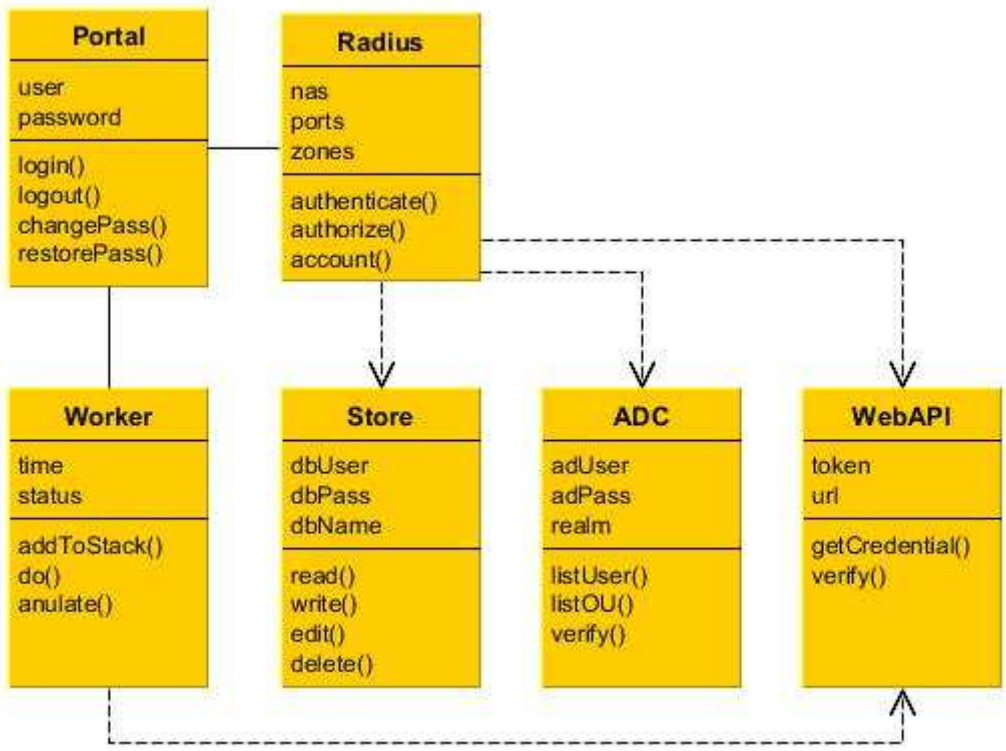

Fig. 5. Diagrama conceptual de clases

La clase Portal representa el portal cautivo y todos los subsistemas que se exponen directamente al usuario final, a través de sus métodos realiza las labores esenciales de la solución, a saber, ingresar y salir del entorno de red que protege el portal; así mismo esta clase permite cambiar y recuperar la contraseña del usuario. La clase portal presenta relaciones de asociación con la clase Radius y la clase Worker, ya que hace uso de los métodos públicos que estas exponen para poder llevar a cabo sus labores objetivo.

La clase Worker ofrece los métodos necesarios para realizar una tarea administrativa en el sistema, a través de su método do() el cual invoca internamente a las subrutinas del sistema que se encargan de cada tarea concreta. Los métodos addToStack() y anulate() permiten registrar o eliminar acciones de la pila de eventos programados, es decir estos métodos dan acceso al servicio CRON y por ende al calendario de tareas programadas.

La clase Radius representa al servidor RADIUS concretamente, siendo este el encargado de realizar todos los procesos de autenticación, autorización y seguimiento del sistema, razón por la cual esta clase expone dichos métodos.

La clase Store representa los medios de almacenamiento persistentes del sistema, sus métodos permiten almacenar, recuperar, actualizar y eliminar información ya sea de la base de datos local o de un servicio remoto.

La clase ADC representa los medios necesarios para colaborar con sistemas LDAP y MS Active Directory, por tanto, engloba las labores realizadas por los subsistemas responsables de gestionar el protocolo LDAP, así como el servidor SAMBA.

La clase WebAPI ofrece el servicio de integración con proveedores de credenciales externos, su método getCredential() permite recuperar el identificador único de usuario en un servicio concreto; su método verify() permite usar el servicio público para identificar a un usuario mediante sus credenciales en dicho servicio.

4.3.2. DIAGRAMA DE BLOQUES

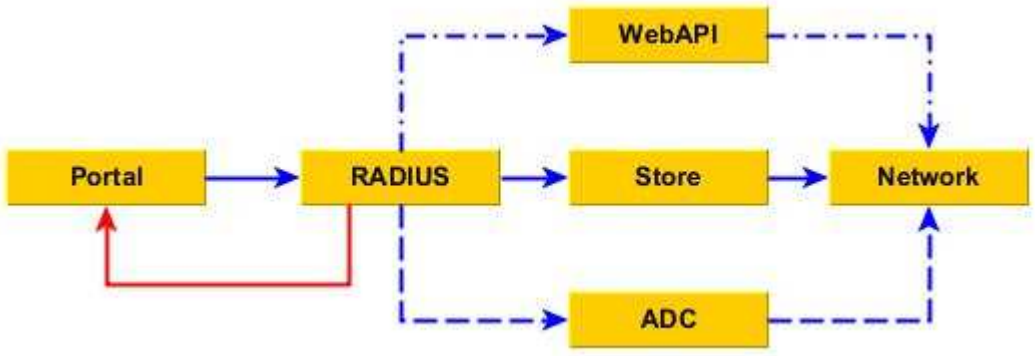

Fig. 6. Diagrama conceptual de bloques

La **Fig. 6** ilustra el proceso maestro de la solución propuesta, es decir, permitir el acceso de un usuario final al entorno de red protegido por el sistema, verificando sus credenciales a través de un protocolo AAA.

La ruta azul del diagrama, representa el proceso exitoso, la roja, implica que se ha presentado un error y el flujo de trabajo debe reiniciarse; esto puede ocurrir por diversos motivos, siendo el más común un ingreso incorrecto de credenciales por parte del usuario.

En primera instancia el flujo pasa por el Portal, donde el usuario ha realizado la solicitud de servicio por cualquiera de los métodos disponibles, esta petición pasa en formato de protocolo RADIUS al siguiente bloque.

El RADIUS en virtud del tipo de credenciales que recibe, escoge cual proveedor de credenciales está en condiciones de verificar la petición, ya sea mediante una credencial local obtenida en el Store, una credencial de directorio activo o una credencial pública que debe verificar por medio del WebAPI; esta es la razón por la cual el flujo azul se divide en tres ramas.

Una vez que RADIUS ha logrado verificar las credenciales, el flujo salta al entorno de red, dándole al usuario la posibilidad de hacer uso de este, bajo las reglas que el sistema RADIUS le haya impuesto.

4.3.3. DIAGRAMAS DE SECUENCIA

Los siguientes diagramas de secuencia ilustran el flujo de procesos que se genera al procesar una petición según cada modelo. Los diagramas solo ilustran el trabajo referente a procesar la petición, no el proceso de configuración previa necesario para llevar a cabo tal labor.

Algunos modelos tienen variantes de forma en su procesamiento; sin embargo, dado que tales variaciones dependen de configuraciones previas y no del flujo de trabajo en sí mismo, éstas se han omitido.

En la **Fig. 7** se muestra un diagrama de secuencia para el portal cautivo con proveedor público.

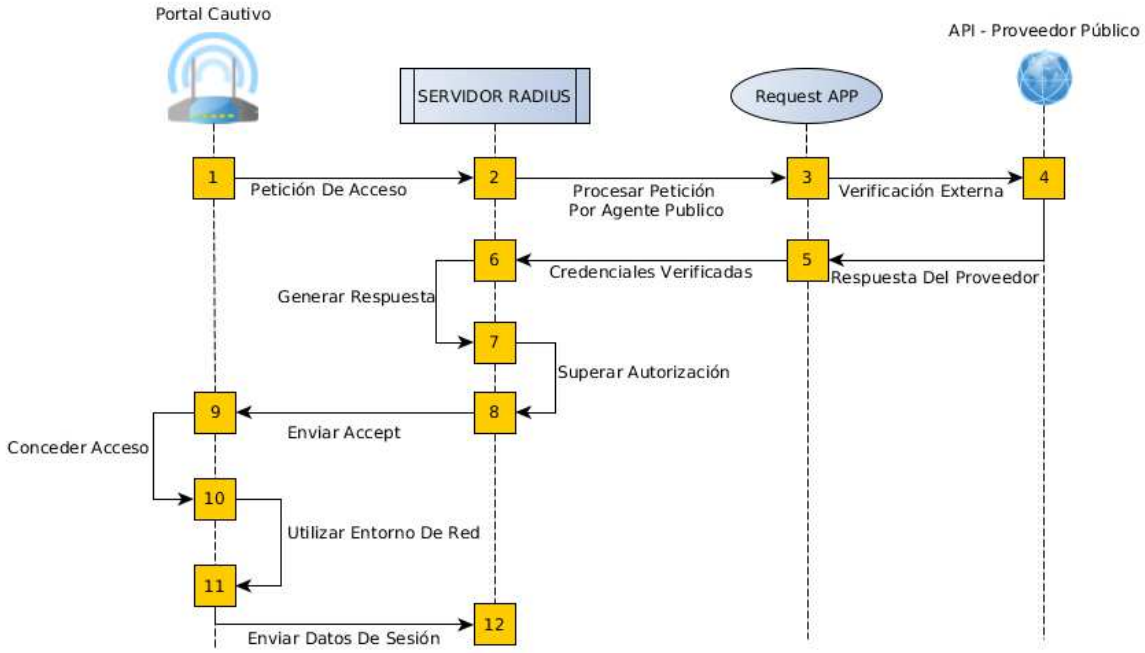

Fig. 7 . Diagrama de secuencia para el portal cautivo con proveedor público

En la **Fig. 8** se muestra un diagrama de secuencia para el portal cautivo con proveedor externo.

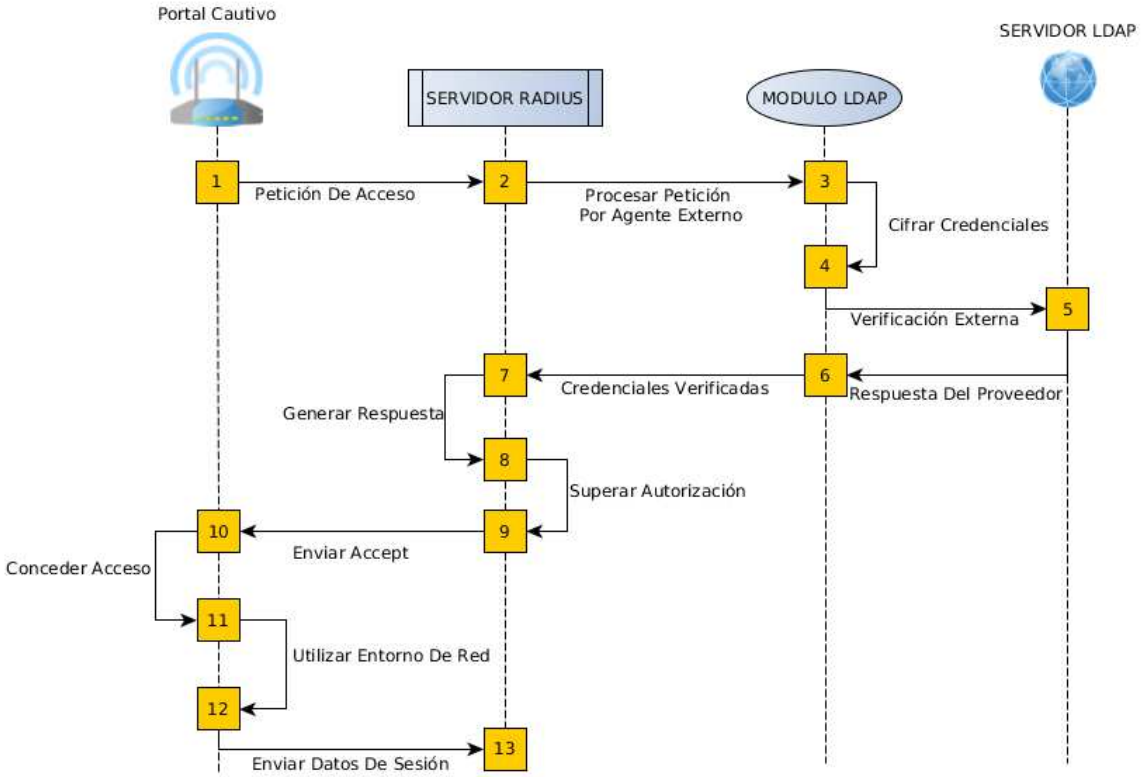

Fig. 8. Diagrama de secuencia para el portal cautivo con proveedor externo

En la **Fig. 9** se muestra un diagrama de secuencia para el portal cautivo con acceso directo sin captura de credenciales de autenticación.

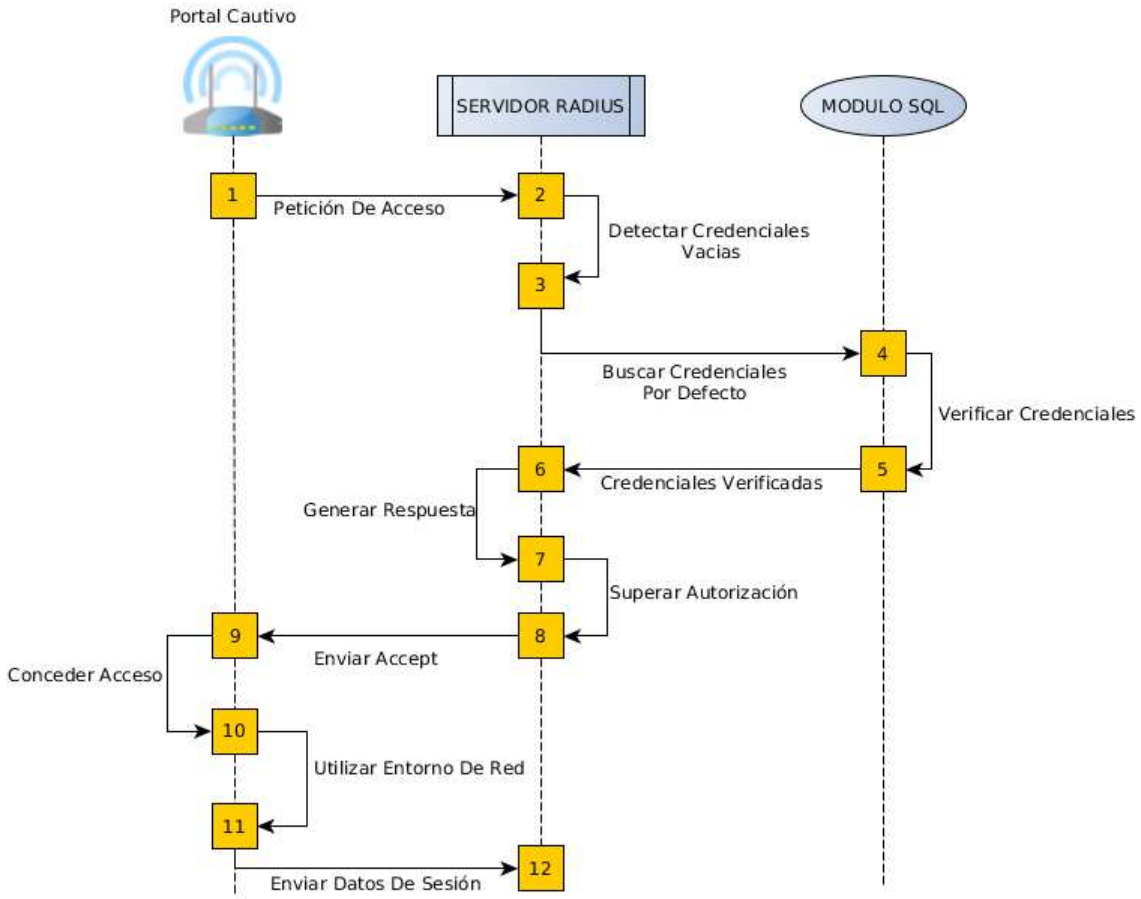

Fig. 9. Diagrama de secuencia para el portal cautivo con acceso directo sin captura de credenciales de autenticación

En la **Fig. 10** se muestra un diagrama de secuencia para el portal cautivo con conexión por medio de credenciales almacenadas.

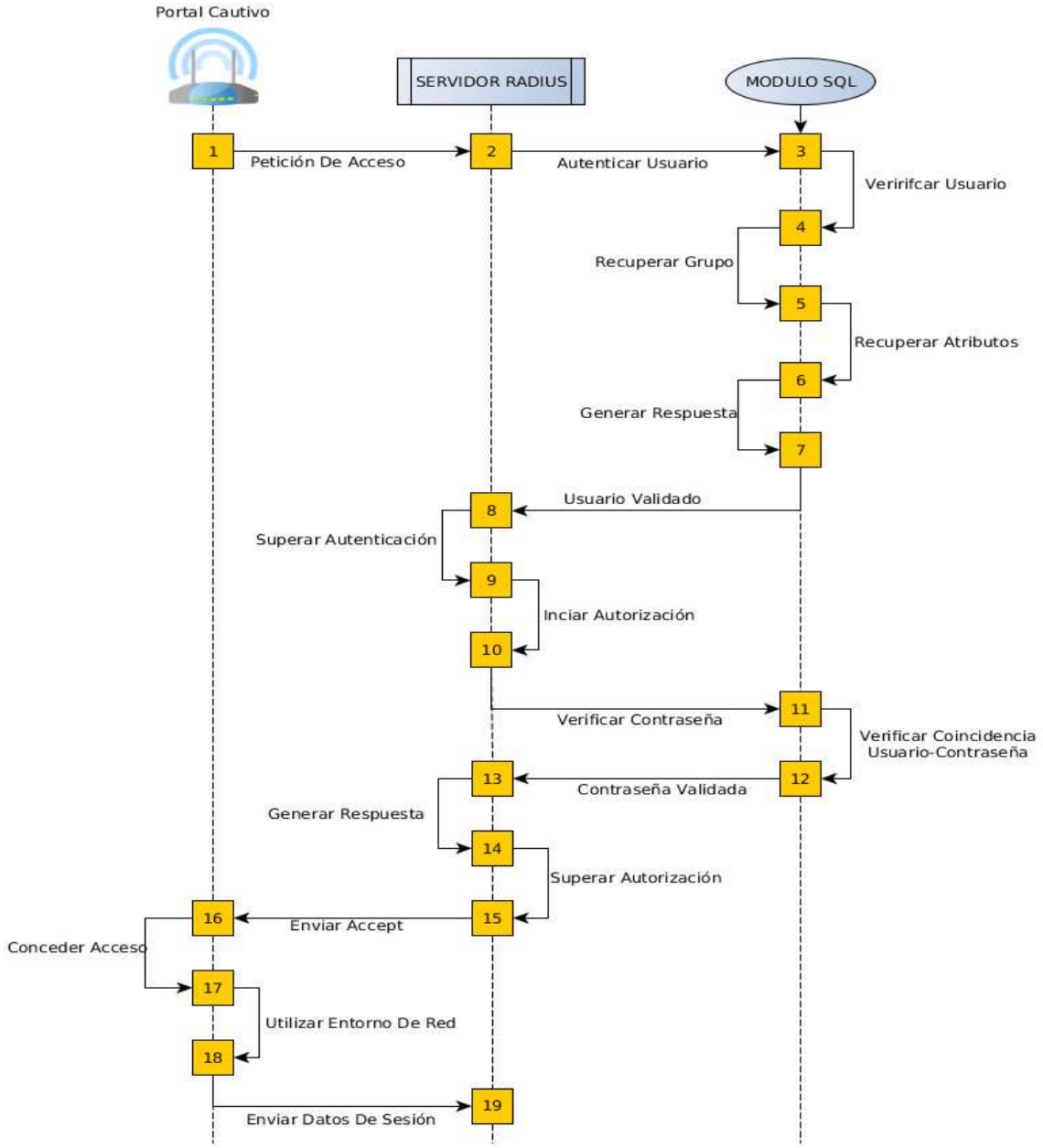

Fig. 10. Diagrama de secuencia para el portal cautivo con credenciales almacenadas

En la **Fig. 11** se muestra un diagrama de secuencia para el portal cautivo para captura de datos por medio de formularios. Una vez concluido este proceso, el flujo de trabajo continúa según lo descrito en la **Fig. 10.**

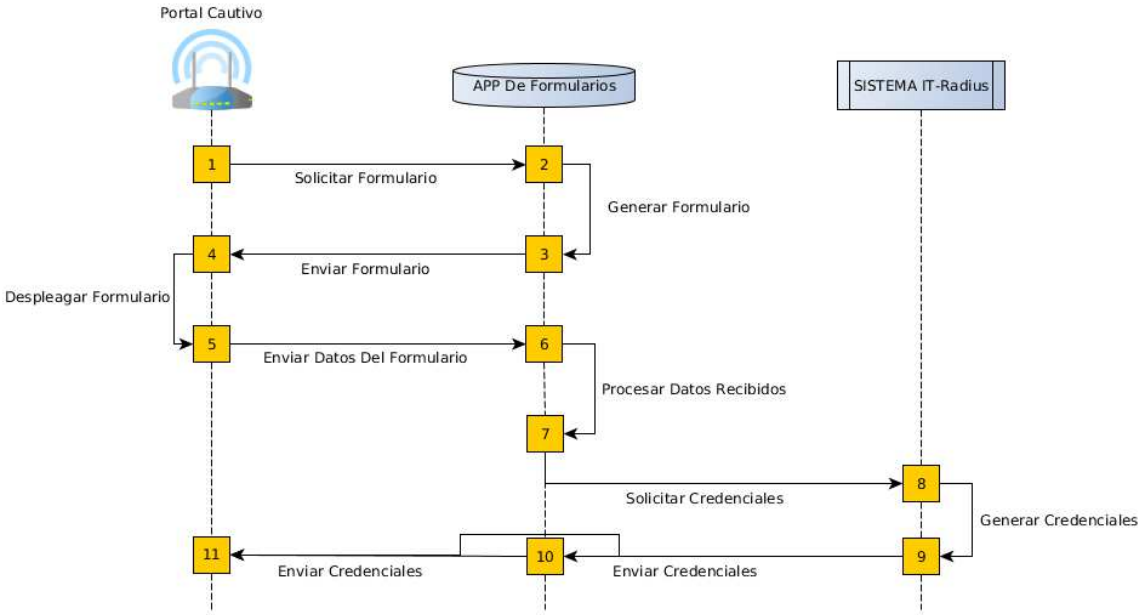

Fig. 11. Diagrama de secuencia para el portal cautivo para captura de datos por medio de formularios.

5. PRUEBAS Y ANÁLISIS DE RESULTADOS

En esta sección se presentará una serie de pruebas para la verificación y validación del sistema.

5.1. PROTOCOLO DE PRUEBAS

A continuación, se describe el protocolo general de pruebas llevado a cabo durante el desarrollo de la solución en su versión inicial.

5.1.1. PLANTEAMIENTO

El presente esquema de pruebas tiene como objetivo fundamental evaluar la respuesta pertinente de la solución ante los diferentes escenarios a los cuales puede ser sometida, teniendo especial consideración en que esta es esencialmente un conjunto de componentes que interactúan para ofrecer un servicio y no una única pieza de software que pueda ser probada de forma puntual y aislada de su entorno.

Si bien la solución es un conjunto tanto de elementos físicos como lógicos, en el presente esquema se asume la idoneidad de los componentes de hardware basándose en la información ofrecida por el fabricante de los mismos. Por tanto, se hará uso del esquema general para pruebas de software, omitiendo aquellas que se consideren irrelevantes para ciertos componentes, ya que la mayor parte de estos son piezas de software desarrolladas por terceros para propósitos generales y que por ende están ampliamente probadas en la mayoría de los escenarios pertinentes.

5.1.2. FASES Y OBJETIVOS

A continuación, se detallan las fases de pruebas a realizar, los objetivos a alcanzar en cada una de ellas y los componentes que se evaluarán.

- Pruebas Unitarias: Se tiene como objetivo evaluar los componentes propios de la solución, a saber: Workers (ver anexo B), Webmin, WebAPI. A fin de revisar su correcto funcionamiento en diversos escenarios.
- Pruebas de integración: Pretenden revisar la integración directa entre los componentes verificados en las pruebas unitarias con los componentes mayores, a saber: Servidor AAA, Portal Cautivo y HostAPD. Así como la correcta ejecución de los procesos integrados entre los mismos.
- Pruebas del sistema: Tienen como objetivo verificar la correcta prestación de los servicios objetivos del sistema, bajo un entorno controlado. Se evaluarán todos los componentes, propios y de terceros con especial énfasis en su comportamiento bajo escenarios adversos.
- Pruebas de despliegue: Presentan el mismo objetivo y características que las pruebas del sistema, salvo que estas se realizarán en un ambiente de producción, bajo un flujo

de operación dictaminado por las actividades de los usuarios, en lugar de en un escenario controlado de laboratorio.

5.1.3. PRUEBAS UNITARIAS

PRUEBA GENERAL DEL WEBMIN	
OBJETIVO	Evaluar la eficacia general del Webmin
FUNCIONES A PROBAR	Configurar servicios, CRUD (Create, Read, Update and Delete) de usuarios administradores, CRUD de usuarios generales, herramientas de testeo.
CASO DE PRUEBA	Según la función a evaluar se crearán una serie de casos, en los cuales se pruebe la respuesta esperada en virtud del proceso realizado. Se crearán como mínimo tres casos de pruebas, uno perfecto según el formato de la función, uno con fallas leves de las cuales el componente debe sobreponerse y uno inaceptable que el componente debe rechazar.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA GENERAL DEL WEB API	
OBJETIVO	Evaluar la eficacia general del Web API
FUNCIONES A PROBAR	Recepción de las peticiones externas, verificación de seguridad, generación de peticiones para servicios externos, procesamiento de resultados.
CASO DE PRUEBA	Según la función a evaluar se realizarán 3 casos concretos, con diferentes niveles de complejidad. Para funciones que dependan de servicios externos, se crearán 10 casos para cada proveedor.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA GENERAL DE LOS WORKERS	
OBJETIVO	Evaluar la eficacia general de los Workers
FUNCIONES A PROBAR	Ejecución de scripts de configuración, ejecución de tareas almacenadas, ejecución de tareas programadas en calendario, ejecución de herramientas de apoyo, ejecución de subrutinas, ejecución de procesos de sincronización.
CASO DE PRUEBA	Según la función a evaluar se realizarán 3 casos de prueba, el primero donde la invocación al worker es perfectamente correcta, uno donde el worker debe sobreponerse a un error y uno donde el worker debe rechazar la ejecución. Se realizará este esquema con cada posible configuración.

METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

5.1.4. PRUEBAS DE INTEGRACIÓN

PRUEBA AAA-PC	
OBJETIVO	Evaluar la correcta integración entre el servidor AAA y el portal cautivo.
FUNCIONES A PROBAR	Autenticación de un usuario, autorización de un usuario, registro de actividades del usuario, aplicación de reglas por atributo directo, aplicación de reglas por grupo.
CASO DE PRUEBA	Se crearán 100 credenciales de usuario, asignadas a 5 grupos de control, cada credencial de usuario tendrá diferentes configuraciones de atributos, así como cada grupo de control. 20 credenciales tendrán una configuración correcta, 20 tendrán atributos con errores menores, 20 tendrán atributos individuales que contradigan los atributos de su grupo, 20 tendrán atributos con errores críticos y 20 tendrán atributos incompatibles críticamente con su grupo. Para cada credencial se realizarán 3 intentos de conexión y navegación.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA AAA-DB	
OBJETIVO	Evaluar la correcta integración entre el servidor AAA y la base de datos
FUNCIONES A PROBAR	Búsqueda y recuperación de credenciales, carga de configuraciones.
CASO DE PRUEBA	Se ejecutará el servidor AAA en modo de pruebas forzándolo a ejecutar cada una de las consultas SQL necesarias para cooperar con la base de datos. Se realizarán 3 pruebas en cada caso, con sentencias correctas, datos ambiguos y sentencias incorrectas.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA AAA-AD	
OBJETIVO	Evaluar la correcta integración entre el servidor AAA y el servidor ADC
FUNCIONES A PROBAR	Recuperación de listas de usuario, recuperación de metadatos, autenticación y autorización de usuarios.
CASO DE PRUEBA	Mediante la herramienta RADTEST y los scripts de apoyo de samba se realizarán 50 pruebas de verificación de credenciales, 30 pruebas de recuperación de listas de usuario y 20 pruebas de recuperación de metadatos. Durante las pruebas se alterarán las condiciones de la red

	que comunica el AAA con el AD, así mismo se añadirán insistencias en los registros DNS y la sincronización de Fecha/Hora por NTP.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA AAA-API	
OBJETIVO	Evaluar la correcta integración entre el servidor AAA y el Web API
FUNCIONES A PROBAR	Ejecución de tareas solicitadas vía API, comprobación de credenciales de proveedores externos.
CASO DE PRUEBA	Se ejecutará el servidor AAA en modo de pruebas, para luego mediante la herramienta PyRequests enviarle peticiones de control mediante el API. Así mismo, mediante la herramienta RADTEST se realizarán pruebas de verificación de credenciales de proveedores externos. Se realizarán 10 pruebas por cada función del API y se verificarán 10 distintas credenciales de los proveedores externos Google, Facebook y Twitter.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA AAA-CONECTORES	
OBJETIVO	Evaluar la correcta integración entre el servidor AAA y los conectores de apoyo.
FUNCIONES A PROBAR	Verificación de credenciales vía ODBC, verificación de credenciales vía OpenLDAP.
CASO DE PRUEBA	Se ejecutará el servidor AAA en modo de pruebas, para luego mediante la herramienta RADTEST realizar pruebas de verificación de credenciales de proveedores accesibles vía ODBC y OpenLDAP. Se realizarán 50 pruebas con credenciales vía ODBC, de motores Oracle, Informix y SQL Server; además de otras 50 pruebas a servidores LDAP.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA WORKERS-ALL	
OBJETIVO	Evaluar la correcta integración entre los Workers y los componentes mayores.
FUNCIONES A PROBAR	Ejecución de las funciones maestras de los Workers, ejecución de tareas programadas.
CASO DE PRUEBA	Se ejecutarán los componentes mayores en modo de pruebas con salida de Log a texto; para luego forzar la ejecución de los Workers maestros. Para cada worker se realizarán 5 pruebas con diferentes niveles de complejidad.

METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

5.1.5. PRUEBAS DEL SISTEMA

PRUEBA DE CONEXIÓN AL AP	
OBJETIVO	Evaluar la estabilidad en el proceso de conexión de diferentes clientes al AP.
FUNCIONES A PROBAR	Conexión, desconexión y reconexión de clientes.
CASO DE PRUEBA	Se escogerán diferentes equipos clientes (PC, LAPTOP, Tablet, etc.) de diferentes fabricantes y en diferentes escenarios de trabajo. Se evaluará la capacidad de los clientes de conectarse de forma estable a la solución, la correcta asignación de direcciones IP y el despliegue automático del portal cautivo. Se realizarán 10 pruebas por equipo cliente representativo.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA DE EFICACIA CON CREDENCIALES LOCALES	
OBJETIVO	Evaluar la eficacia del sistema para atender peticiones.
FUNCIONES A PROBAR	Autenticación, autorización, seguimiento, navegabilidad y gestión de atributos.
CASO DE PRUEBA	Se realizarán pruebas de servicio en base a 100 usuarios registrados localmente, para cada usuario se realizarán 3 intentos, uno correcto, uno con errores manejables y uno inaceptable. Se revisará la correcta asignación de atributos directos y por grupo, así como la capacidad de la solución para interpretar y aplicar los efectos de tales atributos.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA DE EFICACIA CON CREDENCIALES DE ADC	
OBJETIVO	Evaluar la eficacia del sistema para atender peticiones.
FUNCIONES A PROBAR	Autenticación, autorización, seguimiento, navegabilidad y gestión de atributos.
CASO DE PRUEBA	Se realizarán pruebas de servicio con base en 100 usuarios registrados en un controlador de directorio activo, para cada usuario se realizarán 4 intentos, uno correcto, uno con errores manejables, uno inaceptable y uno correcto, pero no registrado en ADC. Se revisará la correcta asignación de atributos directos y por grupo, así como la capacidad de la solución para interpretar y aplicar los efectos de tales atributos.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA DE EFICACIA CON CREDENCIALES DE PROVEEDOR PÚBLICO	
OBJETIVO	Evaluar la eficacia del sistema para atender peticiones.
FUNCIONES A PROBAR	Autenticación, autorización, seguimiento, navegabilidad y gestión de atributos.
CASO DE PRUEBA	Se realizarán pruebas de servicio con base en 120 usuarios registrados en los diferentes proveedores de credenciales públicas (Google, Facebook, y Twitter), para cada usuario se realizarán 4 intentos, uno correcto, uno con errores manejables, uno inaceptable y uno correcto, pero no registrado en el proveedor objetivo. Se revisará la correcta asignación de atributos directos y por grupo, así como la capacidad de la solución para interpretar y aplicar los efectos de tales atributos.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA DE EFICACIA CON CREDENCIALES DE DISPOSITIVO	
OBJETIVO	Evaluar la eficacia del sistema para atender peticiones.
FUNCIONES A PROBAR	Autenticación, autorización, seguimiento, navegabilidad y gestión de atributos.
CASO DE PRUEBA	Se realizarán pruebas de servicio con base en 100 usuarios registrados, que posean dispositivos asociados a sus credenciales, para cada usuario se realizarán 2 intentos, uno con el dispositivo asociado y otro con un dispositivo diferente suplantando al correcto. Se revisará la correcta asignación de atributos directos y por grupo, así como la capacidad de la solución para interpretar y aplicar los efectos de tales atributos.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

PRUEBA GENERAL DE WEBMIN	
OBJETIVO	Evaluar la eficacia de la herramienta unificada de gestión.
FUNCIONES A PROBAR	Gestión de usuarios, gestión de tareas, configuración general, herramientas de apoyo, herramientas de auditoría.
CASO DE PRUEBA	Se realizarán múltiples pruebas manuales a cada una de las prestaciones ofrecidas por el WebMin, con el fin de verificar que estas se ejecuten apropiadamente, aún en situaciones donde la solución se encuentra bajo altos niveles de carga.
METRICA	Se usará una escala relativa entre 0 y 1, la cual representa la relación entre el número de casos exitosos y el total de casos evaluados.

5.1.6. PRUEBAS DE DESPLIEGUE

Se realizaron las mismas pruebas descritas en el apartado anterior, en una institución en donde se instaló la solución. Los detalles de tal proceso se exponen en el apartado "Escenario de pruebas".

5.2. ESCENARIO DE PRUEBAS

Con el fin de validar las capacidades operativas de la solución, se recurrió a un escenario de pruebas masivo donde el sistema se viese sometido a todas las posibles variaciones de trabajo que puedan presentarse, tanto en términos de ejecución, como de integración con la infraestructura corporativa y la manipulación de la solución por diferentes tipos de usuario

5.2.1. OBJETIVOS

- Realizar la instalación y despliegue de la solución sobre una infraestructura informática corporativa.
- Detectar problemas de funcionamiento de la solución.
- Determinar los márgenes de carga y rendimiento de la solución.
- Identificar posibles problemas originados de la interacción de usuario final con el sistema.

5.2.2. EJECUCIÓN

1. Se desplegó una unidad virtual de la solución sobre el servidor de virtualización central de la Universidad Católica de Pereira. Se asignaron recursos de hardware a la máquina virtual de tal forma que emularan idénticamente los recursos presentes en la solución física.
2. La solución se desplegó sobre el entorno de red corporativa mediante la creación de dos redes virtuales; tal que una se conectó al directorio activo de la institución, mientras la otra ofrecía acceso directo a internet.
3. Se habilitaron credenciales para los estudiantes de la institución de tal forma que estos pudiesen acceder mediante credenciales almacenadas dentro de la solución. Para docentes y administrativos se habilitó la autenticación vía directorio activo, mientras que para visitantes se habilitaron los métodos de autenticación por registro y proveedor público.
4. Mediante el sistema de seguimiento del servidor RADIUS, y con la ayuda del personal de informática de la institución, se realizó un seguimiento minucioso de la actividad de los usuarios.
5. Se registró y detalló minuciosamente todos los cambios de configuración que se realizaron sobre la solución y su entorno de red; reportando todos los inconvenientes.

5.2.3. RESULTADOS

- Los usuarios de portal cautivo presentaron problemas de autenticación relacionados con la expiración de sus cuentas.
- El portal cautivo presenta errores de despliegue si el entorno de red no cuenta con un acceso a internet estable.
- El portal cautivo presenta problemas de resolución de DNS.
- Los dominios estáticos y el "Garden" directo que se configuran en el portal cautivo presentan fallas si el servicio de internet se filtra previamente por un proxy.
- El sistema de configuración de los servicios basado en archivos suele corromperse si se realizan cambios masivos.
- La conexión entre el sistema y el directorio activo presenta inestabilidades recurrentes ocasionadas por las políticas de seguridad del AD y por las actualizaciones automáticas del servidor WINS.
- La integración con servicios de correo para recuperar contraseñas locales tiene a fallar si se usan cuentas públicas.

5.2.4. CONTRAMEDIDAS

En base a los resultados obtenidos en el escenario de pruebas, se tomaron las siguientes medidas correctivas:

- Modificar el sistema de gestión de credenciales de tal forma que al expirar no se eliminen, sino que se inhabiliten, notificando previamente al administrador y al cliente.
- Definir un marco mínimo de configuración de red recomendable para el despliegue de la solución.
- Agregar un sistema de actualización automático para las direcciones incluidas en el "Garden".
- Implementar un sistema de almacenamiento temporal en caché para credenciales con el objetivo de evitar la necesidad de estar constantemente conectado al servidor de AD.
- Recomendar el uso de cuentas corporativas para realizar el "Relay" de correos electrónicos, necesario para la recuperación de contraseñas.
- Implementar un sistema de almacenamiento de configuraciones en DB con filtro de cambios para evitar la corrupción de los ficheros maestros de configuración de los servicios.

5.3. PRUEBAS ESPECÍFICAS

5.3.1. PRUEBA PARA EL PORTAL CAUTIVO CON PROVEEDOR PÚBLICO (OpenID)

El esquema de pruebas para este modelo, se representa en la **Fig. 12**.

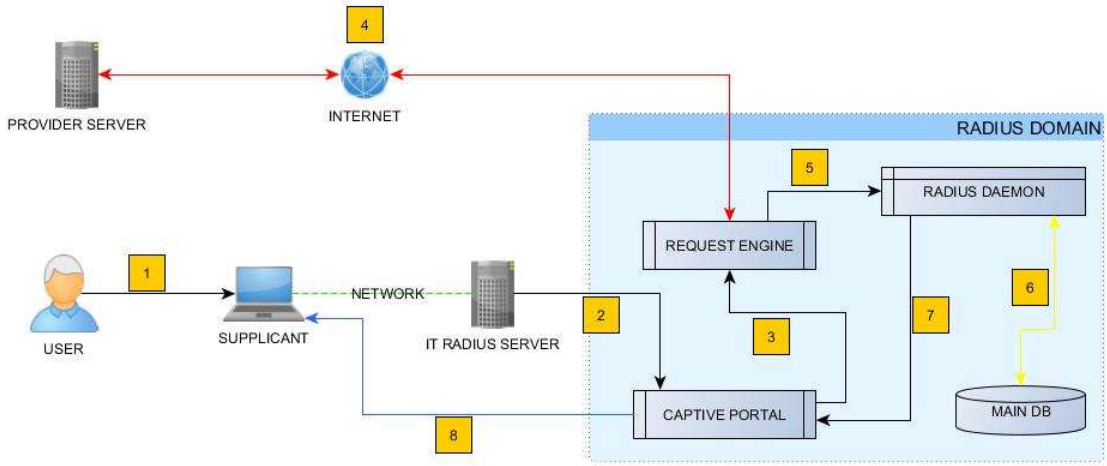

Fig. 12. Prueba para el portal cautivo con proveedor público

Donde se realizan los siguientes procesos:

1. El usuario de prueba accede a la dirección portal cautivo mediante un suplicante (PC, teléfono, etc.); se verifica el acceso de red al portal, así como el correcto despliegue del mismo. El usuario ingresa sus credenciales de acceso de proveedor público disparando el siguiente proceso.
2. La petición llega al servidor RADIUS, ingresando en el dominio del servicio; el portal cautivo recibe las credenciales y la identifica como de tipo público, invocando el siguiente proceso. Se verifica que el portal cautivo reconozca el dominio externo mediante el token [@domain].
3. El motor de peticiones recibe la solicitud del portal cautivo y genera una petición de autenticación pública según el API del proveedor correspondiente. Esta petición es enviada vía internet y se espera la respuesta.

4. Se valida que se reciba una respuesta del proveedor correcto y que esta sea positiva (STATUS 0), en tal caso se inicia el siguiente proceso; si se recibe una respuesta negativa, se ejecuta la siguiente tarea, pero marcando un REJECT como resultado. (ver Fig. 13).

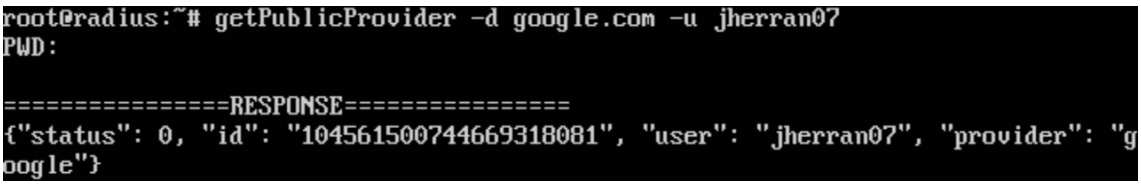

Fig. 13. Ejemplo de solicitud de autenticación sobre un usuario de Google

5. Se ejecuta un proceso de autenticación sobre el demonio RADIUS, de tal manera que este identifique al usuario externo y proceda con las fases de autorización y seguimiento.
6. El demonio RADIUS accede a la base de datos local, extrae los atributos correspondientes al usuario por medio de su grupo(s) asociado(s). Se valida que el demonio RADIUS identifique correctamente el esquema de grupos.
7. El demonio RADIUS emite una respuesta en formato de protocolo RADIUS que es enviada al punto de acceso; este formato es estándar y se aplica indistintamente a todos los modelos de portal cautivo, salvo por algunos atributos que varían según el modelo. Se valida que la respuesta contenga todos los atributos esperados y que sea positiva (ACCEPT) o negativa (REJECT) según sea el caso de la prueba.

Fig. 14. Ejemplo de respuesta en formato de protocolo RADIUS

8. El portal cautivo reconoce la respuesta emitida por el demonio RADIUS y procede a conceder acceso al usuario en caso recibir un indicador positivo; en caso contrario emite los indicadores de error y procede a solicitar nuevamente la autenticación.

5.3.2. PRUEBA PARA EL PORTAL CAUTIVO CON PROVEEDOR EXTERNO

El esquema de pruebas para este modelo, se representa en la **Fig. 15**.

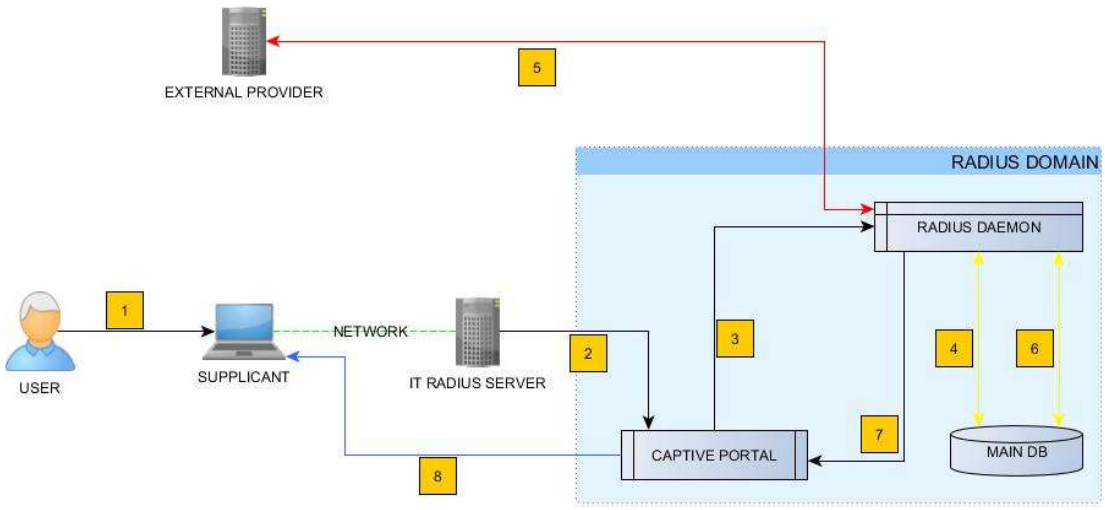

Fig. 15. Prueba para el portal cautivo con proveedor externo

Donde se realizan los siguientes procesos:

1. El usuario de prueba accede a la dirección portal cautivo mediante un suplicante (PC, teléfono, etc.); se verifica el acceso de red al portal, así como el correcto despliegue del mismo. El usuario ingresa sus credenciales de acceso de proveedor externo ejecutando el siguiente proceso.
2. La petición llega al servidor RADIUS, ingresando en el dominio del servicio; el portal cautivo recibe las credenciales y la identifica como de tipo común (para el portal cautivo es indiferente un usuario de proveedor externo a uno local), invocando la siguiente tarea. Se verifica que el portal cautivo determine correctamente el tipo de usuario y genere la petición correspondiente.
3. El demonio de RADIUS recibe la petición del portal cautivo, identifica el tipo de usuario en su fase PRE-AUTH y procede al siguiente proceso. Se verifica que el demonio RADIUS identifique el tipo de usuario como EXTERNO y ejecute el proceso correcto.
4. El demonio RADIUS busca en la base de datos principal, para verificar si posee datos de cache sobre la contraseña del usuario en cuestión. Si es así omite el proceso 5 y salta al 6; en caso contrario realiza el proceso 5. Se valida que el demonio RADIUS envíe la consulta a la base de datos local y reciba la respuesta adecuada.

5. El demonio RADIUS emite una petición al proveedor externo que corresponda al usuario en cuestión, eso lo hace a través de sus módulos auxiliares (LDAP, NTHL, ODBC, PY, etc.); una vez recibida la respuesta del proveedor, si es positiva marca el proceso para devolver un ACCEPT y almacenar la contraseña recibida; en caso contrario marca el proceso para devolver un REJECT. Se valida que el demonio RADIUS identifique correctamente al proveedor externo y que pueda comunicarse satisfactoriamente con él.
6. El demonio de RADIUS accede a la base de datos local, extrae los atributos correspondientes al usuario por medio de su grupo(s) asociado(s). Se valida que el demonio RADIUS identifique correctamente el esquema de grupos.
7. El demonio RADIUS emite una respuesta en formato de protocolo RADIUS que es enviada al portal cautivo; este formato es estándar y se aplica indistintamente a todos los modelos de portal, salvo por algunos atributos que varían según el modelo. Se valida que la respuesta contenga todos los atributos esperados y que sea positiva (ACCEPT) o negativa (REJECT) según sea el caso de la prueba (ver **Fig. 16**.)

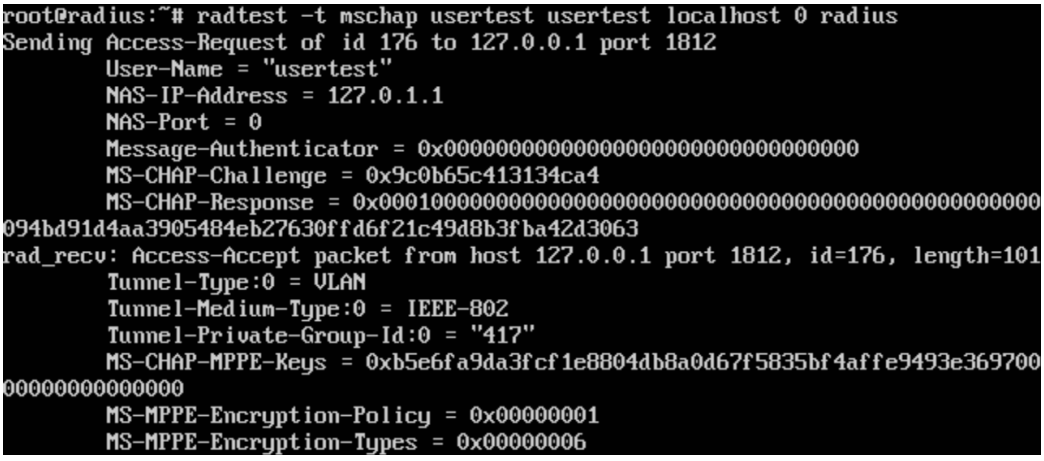

Fig. 16. Ejemplo de respuesta en formato de protocolo RADIUS invocada sobre un proveedor de directorio activo

8. El portal cautivo reconoce la respuesta emitida por el demonio RADIUS y procede a conceder acceso al usuario en caso de recibir un indicador positivo; en caso contrario emite los indicadores de error y procede a solicitar nuevamente la autenticación.

5.3.3. PRUEBA PARA EL PORTAL CAUTIVO CON ACCESO DIRECTO SIN CAPTURA DE CREDENCIALES DE AUTENTICACIÓN

El esquema de pruebas para este modelo, se representa en la **Fig. 17**.

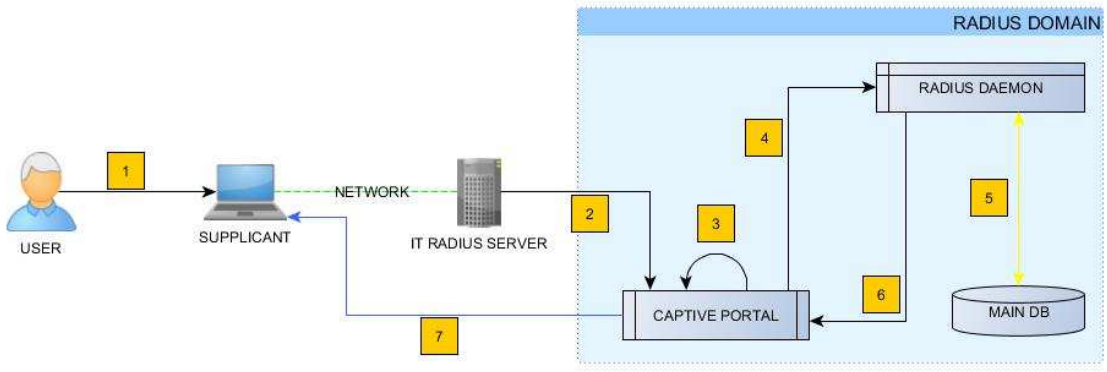

Fig. 17. Prueba para el portal cautivo con acceso directo sin captura de credenciales de autenticación

Donde se realizan los siguientes procesos:

1. El usuario de prueba accede a la dirección portal cautivo mediante un suplicante (PC, teléfono, etc.); se verifica el acceso de red al portal, así como el correcto despliegue del mismo. El usuario no ingresa ningún dato, solo procede a solicitar el acceso al medio, ejecutando el siguiente proceso.
2. La petición llega al servidor RADIUS, ingresando en el dominio del servicio; el portal cautivo recibe las credenciales vacías y la identifica como de tipo genérico, invocando la siguiente tarea. Se verifica que el portal cautivo determine correctamente el tipo de usuario y proceda con el proceso correspondiente.
3. El portal cautivo extrae de su configuración las credenciales del usuario genérico y establece con ellas una petición estándar para el demonio RADIUS. Se valida que el portal cautivo reconozca las credenciales vacías y sea capaz de transformarlas en una petición genérica.
4. El demonio RADIUS recibe la petición del portal cautivo, identifica el tipo de usuario en su fase PRE-AUTH y ejecuta el siguiente proceso. Se verifica que el demonio RADIUS identifique el tipo de usuario como LOCAL y realice el proceso correcto.
5. El demonio RADIUS accede a la base de datos local, extrae los atributos correspondientes al usuario por medio de su grupo(s) asociado(s). Se valida que el demonio RADIUS identifique correctamente el esquema de grupos.
6. El demonio RADIUS emite una respuesta en formato de protocolo RADIUS que es enviada al portal cautivo; este formato es estándar y se aplica indistintamente a todos los modelos de portal, salvo por algunos atributos que varían según el modelo. Se

valida que la respuesta contenga todos los atributos esperados y que sea positiva (ACCEPT) o negativa (REJECT) según sea el caso de la prueba.

7. El portal cautivo reconoce la respuesta emitida por el demonio RADIUS y procede a conceder acceso al usuario en caso de recibir un indicador positivo, de lo contrario emite los indicadores de error y procede a solicitar nuevamente la autenticación.

5.3.4. PRUEBA PARA EL PORTAL CAUTIVO CON CONEXIÓN POR MEDIO DE CREDENCIALES ALMACENADAS

El esquema de pruebas para este modelo, se representa en la **Fig. 18**.

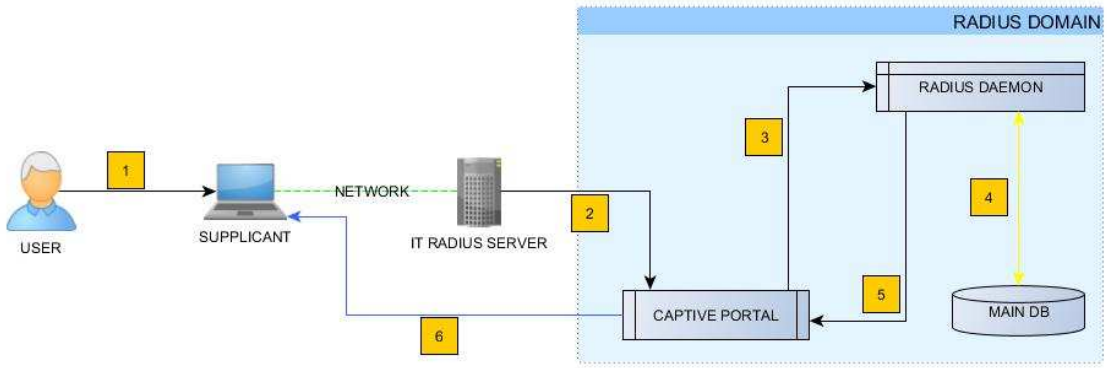

Fig. 18. Prueba para el portal cautivo con conexión por medio de credenciales almacenadas

Donde se realizan los siguientes procesos:

1. El usuario de prueba accede a la dirección portal cautivo mediante un suplicante (PC, teléfono, etc.); se verifica el acceso de red al portal, así como el correcto despliegue del mismo. El usuario ingresa sus credenciales de acceso local, realizando el siguiente proceso.
2. La petición llega al servidor RADIUS, ingresando en el dominio del servicio; el portal cautivo recibe las credenciales vacías y la identifica como de tipo genérico, invocando la siguiente tarea. Se verifica que el portal cautivo determine correctamente el tipo de usuario y desarrolle el proceso correspondiente.
3. El demonio RADIUS recibe la petición del portal cautivo, identifica el tipo de usuario en su fase PRE-AUTH y elabore la siguiente tarea. Se verifica que el demonio RADIUS identifique el tipo de usuario como LOCAL y proceda con el proceso correcto.
4. El demonio RADIUS accede a la base de datos local, extrae los atributos correspondientes al usuario por medio de su grupo(s) asociado(s). Se valida que el demonio RADIUS identifique correctamente el esquema de grupos.

5. El demonio RADIUS emite una respuesta en formato de protocolo RADIUS que es enviada al portal cautivo; este formato es estándar y se aplica indistintamente a todos los modelos de portal, salvo por algunos atributos que varían según el modelo. Se valida que la respuesta contenga todos los atributos esperados y que sea positiva (ACCEPT) o negativa (REJECT) según sea el caso de la prueba.
6. El portal cautivo reconoce la respuesta emitida por el demonio RADIUS y procede a conceder acceso al usuario en caso recibir un indicador positivo de lo contrario emite los indicadores de error y procede a solicitar nuevamente la autenticación.

5.3.5. PRUEBA PARA EL PORTAL CAUTIVO PARA CAPTURA DE DATOS POR MEDIO DE FORMULARIOS.

El esquema de pruebas para este modelo, se representa en la **Fig. 19**.

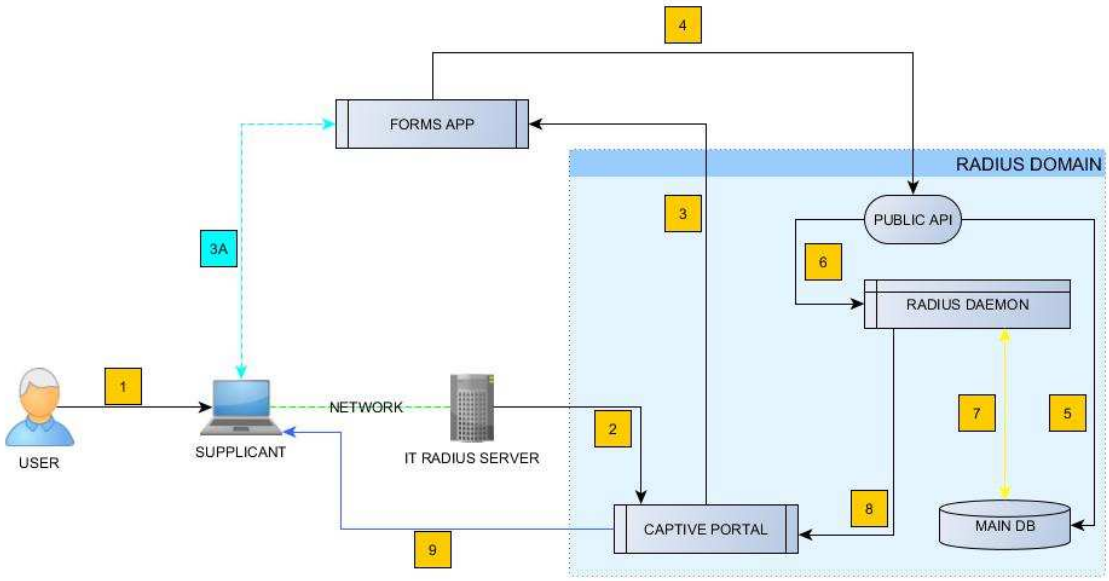

Fig. 19. Prueba para el portal cautivo para captura de datos por medio de formularios

Donde se realizan los siguientes procesos:

1. El usuario de prueba accede a la dirección portal cautivo mediante un suplicante (PC, teléfono, etc.); se verifica el acceso de red al portal, así como el correcto despliegue del mismo. El usuario al no poseer credenciales de acceso invoca el proceso de REGISTRO/CAPTURA DE DATOS.
2. La petición llega al servidor RADIUS, ingresando en el dominio del servicio; el portal cautivo recibe la solicitud de registro y redirección al usuario a la aplicación de formularios; esta aplicación puede ser local o externa en un servidor de red. Se valida

que el portal cautivo identifique la solicitud de registro y que sea capaz de re direccionar el tráfico hacia ella, se verifica la configuración del GARDEN de direcciones y dominios necesarios para que funcione el APP de formularios.

3. El portal cautivo sede el control a la aplicación de formularios, el usuario interactúa directamente con la aplicación de formularios hasta completar su proceso de registro. Se valida la interacción del usuario con el APP de formularios.
4. El APP de formularios se comunica con el dominio RADIUS a través del API publica (**ver anexo A**), registrando los datos y atributos del nuevo usuario. El APP de formularios re direcciona nuevamente al cliente a la zona de espera del portal cautivo. Se valida que el APP de formularios se comunique correctamente con el API pública y que sea capaz de re direccionar al usuario al portal cautivo.
5. Los autómatas del API publica modifican la base de datos local para crear/actualizar los datos y atributos del usuario. Se valida que los autómatas del API publica se ejecuten correctamente y fijen los datos en la base de datos local.
6. Los autómatas del API publica invocan al demonio RADIUS para que ejecute un proceso común, pasándole los datos de usuario que han sido fijados en la base de datos. Se valida que el demonio RADIUS reciba la petición y las credenciales correctamente.
7. El demonio de RADIUS recibe la petición del portal cautivo, identifica el tipo de usuario en su fase PRE-AUTH y efectúa la siguiente tarea. Se verifica que el demonio RADIUS identifique el tipo de usuario como LOCAL y desarrolle el proceso correcto. Posteriormente el demonio de RADIUS accede a la base de datos local, extrae los atributos correspondientes al usuario por medio de su grupo(s) asociado(s). Se valida que el demonio RADIUS identifique correctamente el esquema de grupos.
8. El demonio RADIUS emite una respuesta en formato de protocolo RADIUS que es enviada al portal cautivo; este formato es estándar y se aplica indistintamente a todos los modelos de portal, salvo por algunos atributos que varían según el modelo. Se valida que la respuesta contenga todos los atributos esperados y que sea positiva (ACCEPT) o negativa (REJECT) según sea el caso de la prueba.
9. El portal cautivo reconoce la respuesta emitida por el demonio RADIUS y procede a conceder acceso al usuario en caso de recibir un indicador positivo; en caso contrario emite los indicadores de error y procede a solicitar nuevamente la autenticación.

5.3.6. PROCESO DE VERIFICACIÓN DE EJECUCIÓN DEL SERVIDOR FreeRADIUS 3.X

A continuación, se describe el proceso de verificación de ejecución del servidor FreeRADIUS:

1. Verificar el correcto despliegue del servicio, mediante la ejecución en modo activo y con DEBUG en primer plano. Como súper usuario se ejecuta el comando: **freeradius –s –X**

   Se evalúa la trama de salida del depurador, esperando encontrar el mensaje **"Ready to process request"** (ver **Fig. 20**).

```
radiusd: #### Opening IP addresses and Ports ####
listen {
        type = "auth"
        ipaddr = *
        port = 0
}
listen {
        type = "acct"
        ipaddr = *
        port = 0
}
listen {
        type = "auth"
        ipaddr = 127.0.0.1
        port = 18120
}
 ... adding new socket proxy address * port 51589
Listening on authentication address * port 1812
Listening on accounting address * port 1813
Listening on authentication address 127.0.0.1 port 18120 as server inner-tunnel
Listening on proxy address * port 1814
Ready to process requests.
```

Fig. 20. Segmento final del servidor RADIUS ejecutado en modo de depuración

2. Se corre el servicio FeeRADIUS como demonio maestro del sistema. Como súper usuario se ejecuta el comando: **service freeradius start**
   Se evalúa la respuesta, esperando encontrar el mensaje **"OK"** (ver **Fig. 21**).

```
root@radius:~# service freeradius start
 * Starting FreeRADIUS daemon freeradius                              [ OK ]
```

Fig. 21. Salida del gestor de servicios al iniciar correctamente el servidor RADIUS

3. Se verifica que el proceso de FreeRADIUS este activo en el sistema. Como súper usuario se ejecuta el comando: **ps –A | grep Radius**

   Se espera encontrar al menos un proceso de RADIUS activo (ver **Fig. 22**).

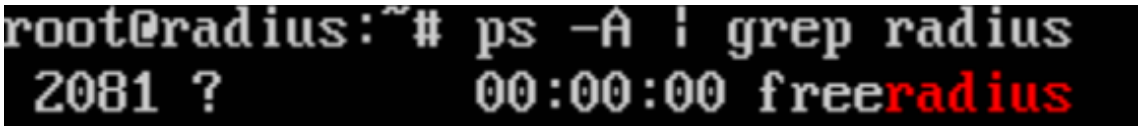

Fig. 22. Salida del monitor de procesos

4. Se verifica la carga y prioridad del servicio RADIUS. Como súper usuario se ejecuta el comando: **top**

   Mediante el teclado se navega por la interfaz de texto hasta encontrar los datos pertenecientes al demonio RADIUS (ver **Fig. 23**).

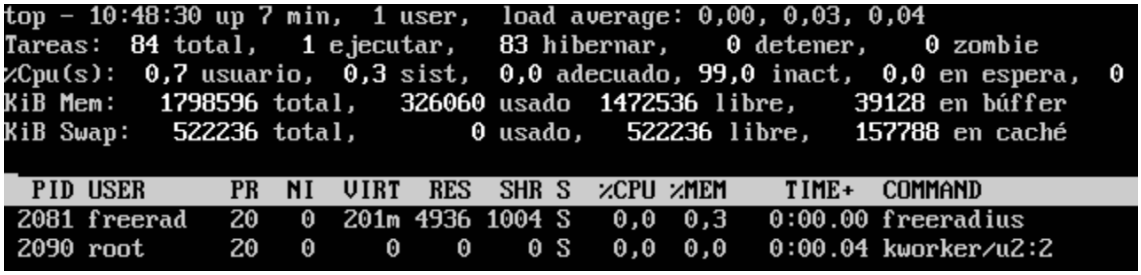

Fig. 23. Salida de la utilidad TOP fijada sobre el demonio del servicio RADIUS

5. Se verifica el estado de los puertos UDP asociados al demonio RADIUS. Como súper usuario se ejecuta el comando: **lsof –i udp | grep rad**

   Se espera encontrar entradas relacionadas con los puertos configurados en el servidor FreeRADIUS (ver **Fig. 24**).

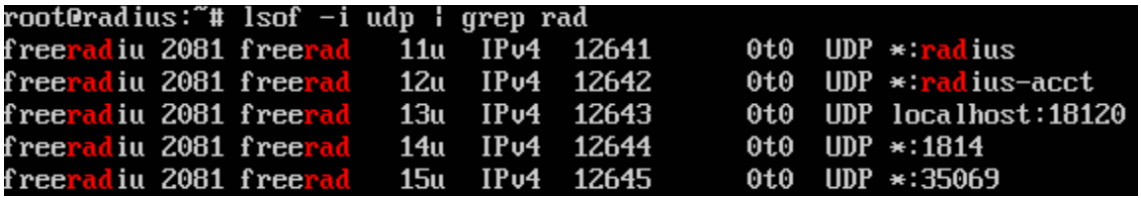

Fig. 24. Puertos configurados en el servidor FreeRADIUS.

5.4. ANÁLISIS Y RESULTADOS

A continuación, se presenta una revisión estadística descriptiva de los resultados obtenidos en las pruebas realizadas.

5.4.1. RESULTADOS DE LAS PRUEBAS UNITARIAS

RESULTADO DE PRUEBA GENERAL DEL WEBMIN			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Crear perfiles o configuraciones	Caso ideal	1	30%
	Caso estándar	0,75	
	Caso crítico	0,5	
Consultar o recuperar perfiles o configuraciones	Caso ideal	1	30%
	Caso estándar	1	
	Caso crítico	0,8	
Actualizar perfiles o configuraciones	Caso ideal	1	10%
	Caso estándar	0,75	
	Caso crítico	0,5	
Anular perfiles o configuraciones	Caso ideal	0,9	20%
	Caso estándar	0,7	
	Caso crítico	0,8	
Generar reportes o estadísticas	Caso ideal	1	10%
	Caso estándar	1	
	Caso crítico	0,8	
RESULTADO GENERAL	**0,83**		**100%**

El resultado general se obtiene como la media ponderada entre el resultado obtenido en cada caso multiplicado por el factor apreciativo de relevancia o impacto. El factor de relevancia o impacto representa qué tan importante es el éxito de ese caso para que se considere que el módulo es eficiente y que puede cumplir con sus funciones. En **Tabla 6** se muestra cómo calcular el resultado general, tomando como ejemplo los resultados de la prueba general del Webmin.

Tabla 6. Ejemplo de cálculo de RESULTADO GENERAL

CASO DE PRUEBA	RESULTADO PROMEDIO		FACTOR DE RELEVANCIA	MULTIPLICAR POR FACTOR DE RELEVANCIA
Crear perfiles o configuraciones	PROMEDIO (1;0,75;0,5)	0,75	0,3	0,225 (0,75*0,3)
Consultar o recuperar perfiles o configuraciones	PROMEDIO (1;1;0,8)	0,93	0,3	0,28 (0,3*0,93)
Actualizar perfiles o configuraciones	PROMEDIO (1;0,75;0,5)	0,75	0,1	0,075 (0,75*0,1)
Anular perfiles o configuraciones	PROMEDIO (0,9;0,7;0,8)	0,8	0,2	0,16 (0,8*0,2)
Generar reportes o estadísticas	PROMEDIO (1;1;0,8)	0,93	0,1	0,093 (0,93*0,1)
RESULTADO GENERAL (Sumatoria de los valores obtenidos al multiplicar por el factor de relevancia 0,225 + 0,28 + 0,075+ 0,16 +0,093)				0,833

En la prueba general del Webmin se tuvieron en cuenta las funciones primordiales que realiza el Webmin (crear, consultar o recuperar, actualizar y anular perfiles o configuraciones, y generar reportes o estadísticas) en tres diferentes escenarios: un caso ideal, en el cual se cumplen todas las condiciones esperadas para que los procesos se lleven a cabo correctamente; un caso estándar, en el cual se pueden admitir ligeros errores, tales como equivocaciones de entrada, fallas del usuario, requerimientos no satisfechos, entre otros; y un caso crítico inaceptable en el cual la suma de los problemas o errores debe desencadenar en que el sistema rechace o bloquee el proceso para evitar inconsistencias en el sistema.

Un caso crítico se alcanza esencialmente cuando la suma de las acciones actuales o anteriores del usuario llevan a punto sin retorno en los cuales o bien el sistema es incapaz de operar o uno de sus procesos entra en conflicto directo con otro; casos típicos de este escenario pueden ser: múltiples servicios corriendo en el mismo puerto, exceso de carga en la pila de servicios, imposibilidad de acceso a un recurso, conflictos de seguridad, entre otros.

Análisis. De los resultados presentados por esta prueba, es posible deducir que en los casos críticos el Webmin tiende a fallar en una proporción relativamente alta, esto debido a que sus labores desencadenan muchos procesos en paralelo, los cuales son difíciles de rastrear y verificar en su totalidad. En el caso relacionado con la anulación de elementos, la proporción de errores se incrementa incluso en casos ideales, ya que al anular un elemento se corre el

riesgo de provocar fallos indirectos por integridad referencial o resolución de dependencias. Aun así, es posible estimar la prueba como exitosa.

RESULTADO DE PRUEBA GENERAL DEL WEB API			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Consumir servicios externos	Complejidad baja	1	60%
	Complejidad media	0,9	
	Complejidad alta	0,7	
Proveer medios de cooperación con servicios externos	Complejidad baja	1	40%
	Complejidad media	0,8	
	Complejidad alta	0,5	
RESULTADO GENERAL	**0,82**		100%

En la prueba general del web API se tuvieron en cuenta las capacidades del sistema para acceder a recursos de proveedores públicos y consumir los servicios que éstos ofrecen. Así mismo, se evaluó la capacidad de la solución de proveer medios por los cuales sistemas externos puedan cooperar o hacer uso de las prestaciones del ecosistema RADIUS.

Análisis. Esta prueba evidencia como el componente web es propenso a errores, salvo en condiciones ideales, principalmente por la necesidad de contar tanto con un entorno de red estable como con un esquema de configuración que no presente limitaciones, además de la absoluta necesidad de que el lado externo se encuentre disponible, y correctamente configurado para para llevar a cabo el proceso. Es posible considerar la prueba como exitosa ya que se cumple de forma estable y en mayor medida con los objetivos de los casos planteados.

RESULTADO PRUEBA GENERAL DE LOS WORKERS			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Ejecución de scripts de configuración	Caso ideal	1	40%
	Caso estándar	0,8	
	Caso crítico	0,5	
Ejecución de tareas almacenadas	Caso ideal	1	15%
	Caso estándar	0,8	
	Caso crítico	0,7	
Ejecución de tareas programadas en calendario	Caso ideal	1	10%
	Caso estándar	0,8	
	Caso crítico	0,7	
Ejecución de herramientas de apoyo	Caso ideal	1	5%
	Caso estándar	1	
	Caso crítico	1	
Ejecución de subrutinas	Caso ideal	1	20%
	Caso estándar	0,7	
	Caso crítico	0,7	

Ejecución de procesos de sincronización	Caso ideal	0,9	10%
	Caso estándar	0,6	
	Caso crítico	1	
RESULTADO GENERAL	**0,80**		**100%**

En la prueba general de los workers se evaluaron individualmente todos los componentes intermedios que se encargan de mantener sincronizado el ecosistema RADIUS, logrando con ello proveer las funciones básicas y de valor agregado

Análisis. Los resultados de esta prueba evidencian que si bien existen fallas en la ejecución de los workers, producto de la inevitable complejidad de los mismos, éstos son en mayor medida estables, salvo por los escenarios de caso crítico. Dado el gran volumen de elementos que componen los workers y la eminente entropía degenerada por sus labores, es posible considerar a la prueba exitosa, más halla los fallos que se presentan.

Es igualmente importante mencionar que en casos donde los workers dependen de elementos externos, el escenario crítico no suele fallar nunca, ya que el comportamiento esperado en tales circunstancias es que el proceso se aborte para evitar consecuencias negativas, y dado que en el escenario crítico el recurso externo no está disponible, resulta natural que se dé el comportamiento esperado de forma sencilla.

5.4.2. RESULTADOS DE PRUEBAS DE INTEGRACIÓN

RESULTADO DE PRUEBA AAA-PC			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Autenticación y autorización de un usuario	Caso ideal	1	50%
	Caso estándar	0,9	
	Caso crítico	1	
Registro de actividades del usuario	Caso ideal	1	20%
	Caso estándar	1	
	Caso crítico	0	
Aplicación de reglas por atributo	Caso ideal	1	30%
	Caso estándar	0,7	
	Caso crítico	0,5	
RESULTADO GENERAL	**0,83**		**100%**

En la prueba AAA-PC se evaluó la capacidad de cooperación entre el servidor de protocolo RADIUS y el servicio de portal cautivo, tanto en sus funciones básicas de autenticación, autorización y seguimiento como en su capacidad de ofrecer servicios enriquecidos a través del sistema de usuarios, grupos y atributos.

Análisis. Los resultados de esta prueba permiten extraer tres conclusiones primordiales, a saber:

1. El proceso de autenticación y autorización se comporta de forma perfecta en el escenario óptimo, ya que todos los elementos se encuentran disponibles alcanzando, por tanto, el resultado esperado. Así mismo, en el escenario crítico el comportamiento es perfecto, ya que el par externo no se encuentra disponible y se alcanza un estado de rechazo en la petición, lo cual es precisamente el resultado esperado en tales circunstancias.

2. En el caso de registro de actividades, bajo el escenario crítico se obtiene el peor resultado posible, esto debido a que la prueba está planteada de tal forma que en condiciones críticas el registro debería detenerse, sin embargo, dada la configuración base del servidor RADIUS el proceso de registro es universal y no se puede discriminar su ejecución por escenario o usuario.

3. La aplicación de reglas por atributo presentan un elevado índice de fallas en condiciones críticas, dado que el protocolo RADIUS no define ningún mecanismo para priorizar atributos cuando un perfil presenta redundancias en sus definiciones, por lo que la solución a esto se deja completamente al azar.

Bajo la perspectiva anterior es posible considerar a la prueba como exitosa.

RESULTADO DE PRUEBA AAA-DB			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Búsqueda y recuperación de credenciales	Caso ideal	1	50%
	Caso estándar	0,9	
	Caso crítico	0,7	
Carga de configuraciones	Caso ideal	1	50%
	Caso estándar	0,8	
	Caso crítico	0,6	
RESULTADO GENERAL	**0,83**		**100%**

En la prueba AAA-DB se evalúa la capacidad del servidor de protocolo RADIUS de interactuar con un proveedor de base de datos relacionales para realizar las funciones de manejo de información vitales para la ejecución del ecosistema.

Análisis. Los resultados de esta prueba, evidencian claramente como en casos críticos, donde la configuración del motor RDBMS no satisface los requisitos del servidor RADIUS, se incrementan notablemente los errores de la solución, especialmente en las labores relacionadas con la carga dinámica de configuraciones, lo que a su vez puede degenerar en problemas catastróficos. Sin embargo, dado que los escenarios críticos son muy escasos salvo que se produzcan intencionadamente, es posible considerar esta prueba como exitosa.

RESULTADO PRUEBA AAA-AD			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Recuperación de listas de usuario	Caso ideal	0,8	25%
	Caso estándar	0,7	
	Caso crítico	1	
Recuperación de metadatos	Caso ideal	0,8	15%
	Caso estándar	0,7	
	Caso crítico	1	
Autenticación y autorización de usuarios	Caso ideal	0,9	60%
	Caso estándar	0,75	
	Caso crítico	1	
RESULTADO GENERAL	**0,86**		**100%**

La prueba AAA-AD tiene como objetivo evaluar el rendimiento del componente RADIUS enlazado a un controlador de directorio activo como fuente de credenciales, explorando con ello las diferentes complicaciones que pueden originarse en tal escenario.

Análisis. En primer lugar, es importante recalcar que la solución opera perfectamente en condiciones críticas, ya que está diseñada para rechazar todo tipo de peticiones si el par remoto no se encuentra disponible, lo cual es el comportamiento deseado para tales circunstancias. De igual forma, es importante mencionar que en escenarios comunes el rendimiento presentado es claramente mediocre, esto principalmente debido a la inestabilidad inherente que existe al acoplar un controlador AD con un servidor base Linux. A pesar de que la prueba puede considerarse como exitosas, es necesario concluir que la conexión directa AAA-AD no es ideal y debe ser mejorada.

RESULTADO DE PRUEBA AAA-API			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Ejecución de tareas solicitadas vía API	Caso ideal	1	40%
	Caso estándar	0,9	
	Caso crítico	0,7	
Comprobación de credenciales de proveedores externos.	Caso ideal	1	60%
	Caso estándar	0,8	
	Caso crítico	1	
RESULTADO GENERAL	**0,90**		**100%**

La prueba AAA-API permite evaluar la correcta sincronización entre el tráfico gestionado mediante el API y sus efectos sobre el servidor RADIUS base, tanto desde la perspectiva del consumidor de recursos remotos como desde el enfoque de servidor público desligado.

Análisis. Los resultados de esta prueba remarcan como la solución está preparada para proceder correctamente en escenarios críticos donde la colaboración con elementos externos

es fundamental y estos no se encuentran disponibles. Así mismo, es apropiado estimar que el enfoque como prestador de recursos en condiciones ideales y normales es casi perfecto, decayendo en eficiencia solo en escenarios críticos donde el entorno de red es altamente desfavorable.

RESULTADO DE PRUEBA AAA-CONECTORES			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Verificación de credenciales vía ODBC	Caso ideal	0,9	50%
	Caso estándar	0,8	
	Caso crítico	1	
Verificación de credenciales vía OpenLDAP.	Caso ideal	1	50%
	Caso estándar	0,9	
	Caso crítico	1	
RESULTADO GENERAL	**0,93**		**100%**

El objetivo de la prueba AAA-CONECTORES es evaluar el rendimiento del sistema RADIUS cuando trabaja con fuentes externas como proveedores de credenciales, siempre y cuando la conexión con estos se realice bajo protocolos y herramientas de libre acceso.

Análisis. Esta prueba puede considerarse altamente exitosa tanto por sus resultados directos como al compararla con la prueba AAA-AD, la cual es muy similar en concepto, pero radicalmente diferente en ejecución, dada la naturaleza cerrada del protocolo y las tecnologías que esta precisa. Teniendo esto en cuenta es posible concluir sintéticamente que el sistema RADIUS es más estable y eficiente cuando trabaja bajo esquemas de conexión abiertos que cuando debe operar bajo esquemas privativos.

RESULTADO PRUEBA WORKERS-ALL			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Ejecución de las funciones maestras de los Workers	Caso ideal	0,9	80%
	Caso estándar	0,7	
	Caso crítico	0,5	
Ejecución de tareas programadas.	Caso ideal	1	20%
	Caso estándar	0,8	
	Caso crítico	0,8	
RESULTADO GENERAL	**0,73**		**100%**

El objetivo primordial de la prueba WORKERS-ALL es avaluar la sinergia que debe existir entre los Workers y los servicios fundamentales del ecosistema RADIUS, dado que de su correcta integración depende la estabilidad y eficiencia general de la solución.

Análisis. En términos generales es posible estimar esta prueba como exitosa, sin embargo, es preciso mencionar que en condiciones críticas y muchos escenarios normales, la complejidad y la entropía que desencadenan los Workers maestros tiende a provocar errores, la mayoría

de esos aleatorios; por lo que es necesario concluir que el esquema Workers debe sr revisado y simplificado a fin de mejorar la estabilidad general del sistema.

5.4.3. RESULTADOS DE PRUEBAS DEL SISTEMA

RESULTADO DE PRUEBA DE CONEXIÓN AL AP			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Conexión, desconexión y reconexión de clientes.	Caso ideal	1	100%
	Caso estándar	0,8	
	Caso crítico	0,5	
RESULTADO GENERAL	**0,76**		**100%**

La prueba de conexión al AP tiene como objetivo evaluar el rendimiento de la solución en escenarios donde las condiciones físicas del entorno afectan directamente a la experiencia de usuario.

Análisis. Es posible considerar esta prueba como exitosa, ya que, si bien los resultados en escenarios críticos son mediocres, esto se debe a imposibilidades físicas insorteables, salvo mediante modificaciones de hardware que están fuera de los alcances del proyecto.

RESULTADO DE PRUEBA DE EFICACIA CON CREDENCIALES LOCALES			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Autenticación, autorización, seguimiento, navegabilidad y gestión de atributos.	Caso ideal	1	100%
	Caso estándar	0,9	
	Caso crítico	0,6	
RESULTADO GENERAL	**0,83**		**100%**

El objetivo de esta prueba es evaluar la estabilidad y rendimiento general de la solución, cuando se usan únicamente los recursos locales como proveedores de credenciales.

Análisis. Esta prueba puede considerarse altamente exitosa, dados los resultados tanto en escenarios ideales como generales, aun así, es preciso mencionar que bajo condiciones críticas queda un amplio margen de mejora para futuras revisiones de la solución.

RESULTADO PRUEBA DE EFICACIA CON CREDENCIALES DE ADC			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Autenticación, autorización, seguimiento, navegabilidad y gestión de atributos.	Caso ideal	1	100%
	Caso estándar	0,8	
	Caso crítico	1	
RESULTADO GENERAL	**0,93**		**100%**

La prueba de eficacia con credenciales de ADC tiene como objetivo evaluar la estabilidad y el rendimiento del sistema RADIUS cuando se usa un controlador de directorio activo como único proveedor de credenciales.

Análisis. Los resultados de esta prueba pueden considerarse como contradictorios si se contrastan con los de la prueba AAA-AD, sin embargo, tales diferencias pueden explicarse si se tiene en cuenta que en esta ocasión la prueba se basa en medir la eficacia de la solución, es decir su capacidad de realizar las labores que se esperan de ella, y que para tal fin se tiene en cuenta el resultado final que ofrece todo el ecosistema, en lugar de únicamente el arrojado por el núcleo RADIUS como era el caso en la prueba anterior. Una vez explicada la discrepancia en los resultados es posible considerar esta prueba como exitosa.

RESULTADO DE PRUEBA DE EFICACIA CON CREDENCIALES DE PROVEEDOR PÚBLICO			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Autenticación, autorización, seguimiento, navegabilidad y gestión de atributos.	Caso ideal	1	100%
	Caso estándar	0,9	
	Caso crítico	1	
RESULTADO GENERAL	**0,96**		**100%**

La prueba de eficacia con credenciales de proveedor público como objetivo evaluar la estabilidad y el rendimiento del sistema RADIUS cuando se usan servicios públicos como única fuente de credenciales.

Análisis. Gracias a los componentes de apoyo que le permiten al ecosistema RADIUS colaborar de forma trasparente con los principales proveedores públicos de credenciales, es posible considerar esta prueba como altamente exitosa, ya que solo bajo escenarios muy puntuales donde el entorno de red es inestables, aleatorio y adverso, se presentan resultados no deseados.

RESULTADO DE PRUEBA DE EFICACIA CON CREDENCIALES DE DISPOSITIVO			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Autenticación, autorización, seguimiento, navegabilidad y gestión de atributos.	Caso ideal	1	100%
	Caso estándar	0,8	
	Caso crítico	0,6	
RESULTADO GENERAL	**0,8**		**100%**

La prueba de eficacia con credenciales de dispositivo tiene como objetivo evaluar la estabilidad y el rendimiento del sistema RADIUS cuando se usan las direcciones físicas de los dispositivos clientes como credencial de autenticación.

Análisis. Si bien es cierto que los resultados de la prueba permiten catalogar a esta como satisfactoria, es igualmente cierto que se observa un margen de mejora considerable, esto se debe principalmente a que el proceso de autenticación por dispositivo no es trasparente para el usuario final, sino que implica un proceso de registro previo el cual puede fallar por desconocimiento del usuario sobre los conceptos mínimos necesarios para completar tal labor. También es preciso tener en cuenta que este método depende ampliamente de que los dispositivos clientes ofrezcan verídicamente su dirección física durante el proceso de autenticación y autorización, algo que, si bien suele darse por sentado, en la realidad está bastante alejado del ideal esperado, ya sea por negligencia de los fabricantes que comercializan dispositivos con direcciones clonadas, o por la presencia de software malicioso en el dispositivo el cual puede alterar la firma física del mismo.

RESULTADO DE PRUEBA GENERAL DE WEBMIN			
CASO DE PRUEBA	**RESULTADO**		**RELEVANCIA O IMPACTO**
Gestión de usuarios	Caso ideal	1	30%
	Caso estándar	0,9	
	Caso crítico	0,7	
Gestión de tareas	Caso ideal	1	20%
	Caso estándar	0,8	
	Caso crítico	0,8	
Configuración general	Caso ideal	1	30%
	Caso estándar	0,8	
	Caso crítico	0,6	
Herramientas de apoyo	Caso ideal	1	10%
	Caso estándar	1	
	Caso crítico	1	
Herramientas de auditoría.	Caso ideal	1	10%
	Caso estándar	0,8	
	Caso crítico	0,8	
RESULTADO GENERAL	**0,86**		**100%**

La prueba general de Webmin tiene como objetivo evaluar el Webmin como componente sistémico vital de la solución, ya no centrándose en sus funciones atómicas sino en las prestaciones y responsabilidades generales que recaen sobre él.

Análisis. En primer lugar, es posible estimar los resultados de la prueba como exitosas, no ideales, pero sí satisfactorios. Sin embargo, aunque en términos generales todos los aspectos evaluados se encuentran dentro de lo esperado, esta prueba evidencia claramente como la interacción del usuario humano afecta drásticamente los resultados de un proceso que puede considerar infalible. Tal efecto se debe principalmente al desconocimiento del usuario tanto de conocimientos esenciales sobre informática, como de detalles específicos propios del diseño de la solución o la interfaz del Webmin. Por tanto, es necesario concluir que el Webmin debe ser revisado para hacerlo más "amable" con usuario final, y recomendar

especialmente que las funciones complejas de la solución se encarguen solo a usuarios con un nivel de conocimiento técnico apropiado.

5.4.4. RELACIÓN GENERAL DE ERRORES POR TIPO

Tabla 7. Relación general de errores por tipo

TIPO ERROR	CANTIDAD	PORCENTAJE
LOGICO DE PROGRAMACIÓN	231	8,1
DE ENTORNO DE RED	417	14,5
DE CONFIGURACIÓN	129	4,5
DE INTEGRACIÓN CON AD	239	8,3
DE INTEGRACIÓN EXTERNA	69	2,4
DE MANIPULACIÓN DE USUARIO FINAL	1781	62,1

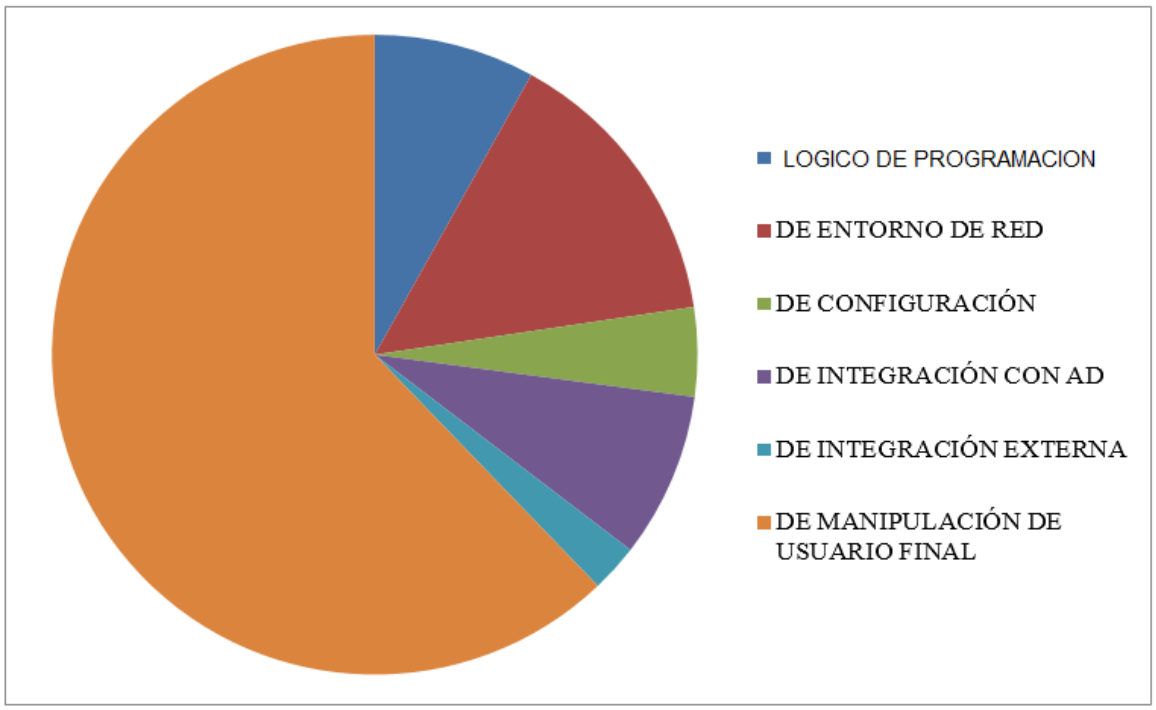

Fig. 25. Relación general de errores por tipo

5.4.5. RELACIÓN DE ERRORES LÓGICOS DE PROGRAMACIÓN

Tabla 8. Relación de errores lógicos de programación.

CAUSA	POCENTAJE

PLANTILLA DE CONFIGURACIÓN INCORRECTA	15,3
CASO ESPECIAL OMITIDO	10,2
MANEJO INCORRECTO DE PROTOCOLO	25,4
CONFIGURACIÓN DE INTEGRACIÓN ERRONEA	18,9
CONFIGURACIÓN DE EJECUCIÓN ERRONEA	11,7
ERRORES GENERALES DE LOGICA	18,5

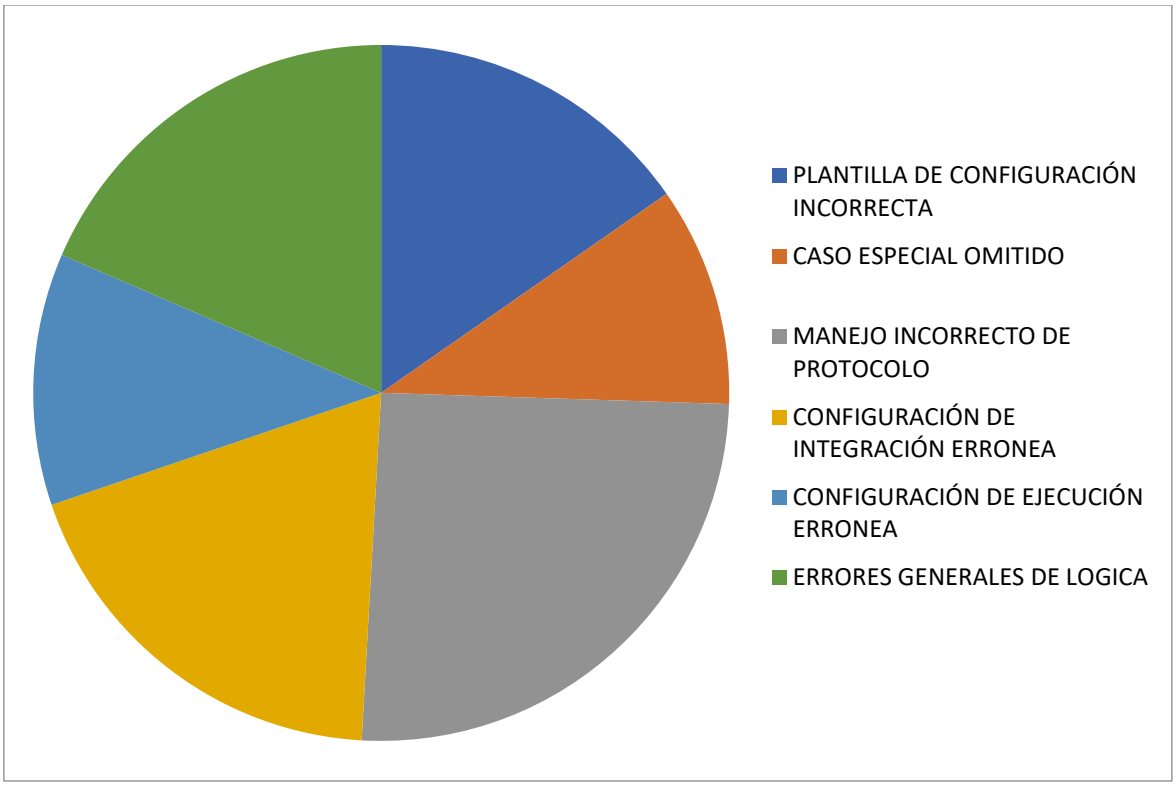

Fig. 26. Relación de errores lógicos de programación

5.4.6. RELACIÓN DE ERRORES OCASIONADOS POR EL ENTORNO DE RED

Tabla 9. Relación de errores ocasionados por el entorno de red.

CAUSA	POCENTAJE
RUTAS ESTATICAS MAL CONFIGURADAS	33,8
POLITICAS DE SEGURIDAD DE RED INCOMPATIBLES	40,2
CONFIGURACIÓN IP/VLAN INCORRECTA	15,9
MEDIO DE TRASMISIÓN DEFECTUOSO	10,1

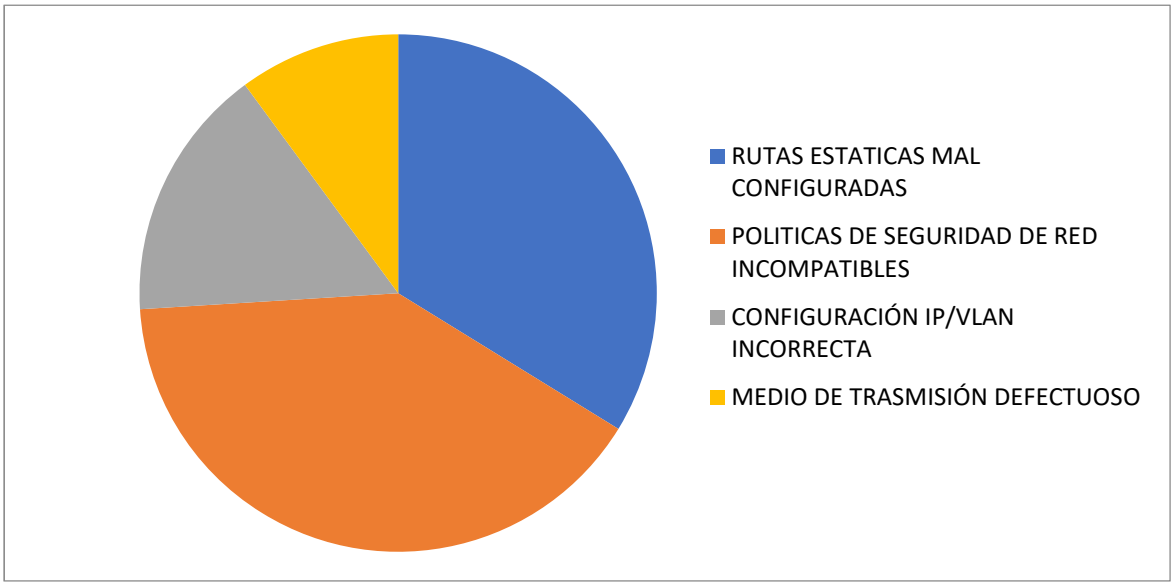

Fig. 27. Relación de errores ocasionados por el entorno de red

5.4.7. RELACIÓN DE CAUSAS PROBABLES DE LOS ERRORES OCASIONADOS POR INTERVENCIÓN HUMANA.

Tabla 10. Relación de causas probables de los errores ocasionados por intervención humana.

CAUSA	PORCENTAJE
FALTA DE DOCUMENTACIÓN PARA CONSULTA	48,0
FALTA DE CONOCIMIENTO TECNICO PARA MANIPULAR LA SOLUCIÓN	15,5
FALTA DE CAPACITACIÓN GENERAL DE LOS USUARIOS FINALES	32,8
INTERFAZ DE USUARIO FINAL POCO INTUITIVA	3,7

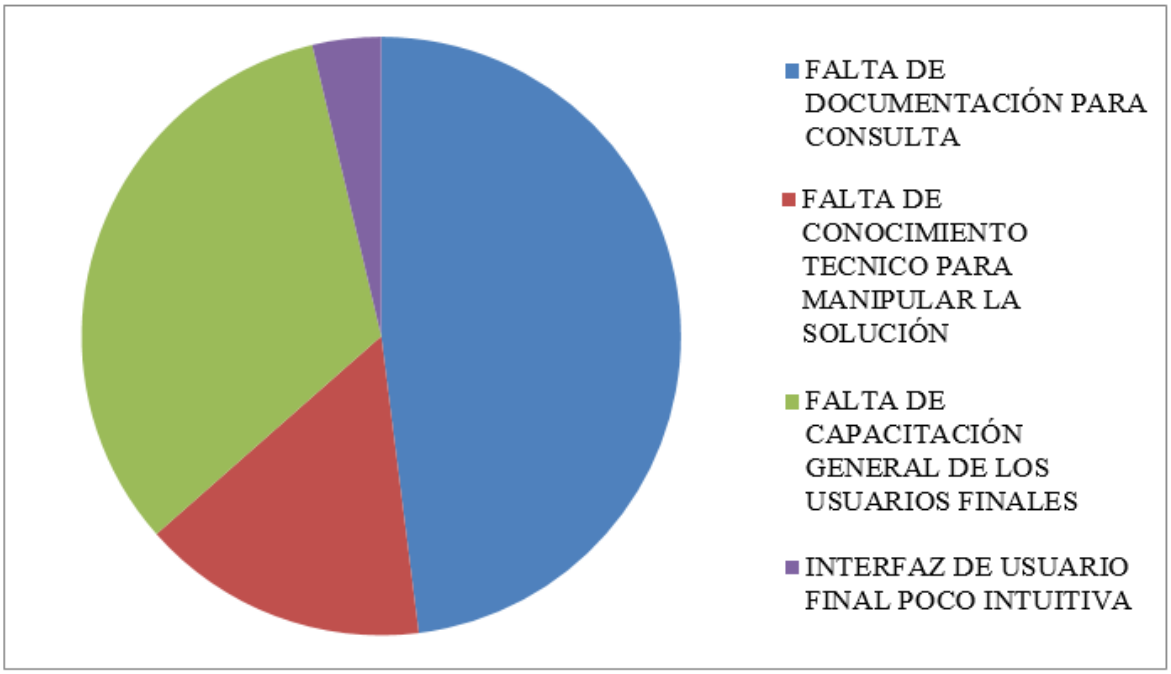

Fig. 28. Relación de causas probables de los errores ocasionados por intervención humana

5.4.8. RELACIÓN USUARIOS CONCURRENTES – ERRORES DE USUARIO FINAL

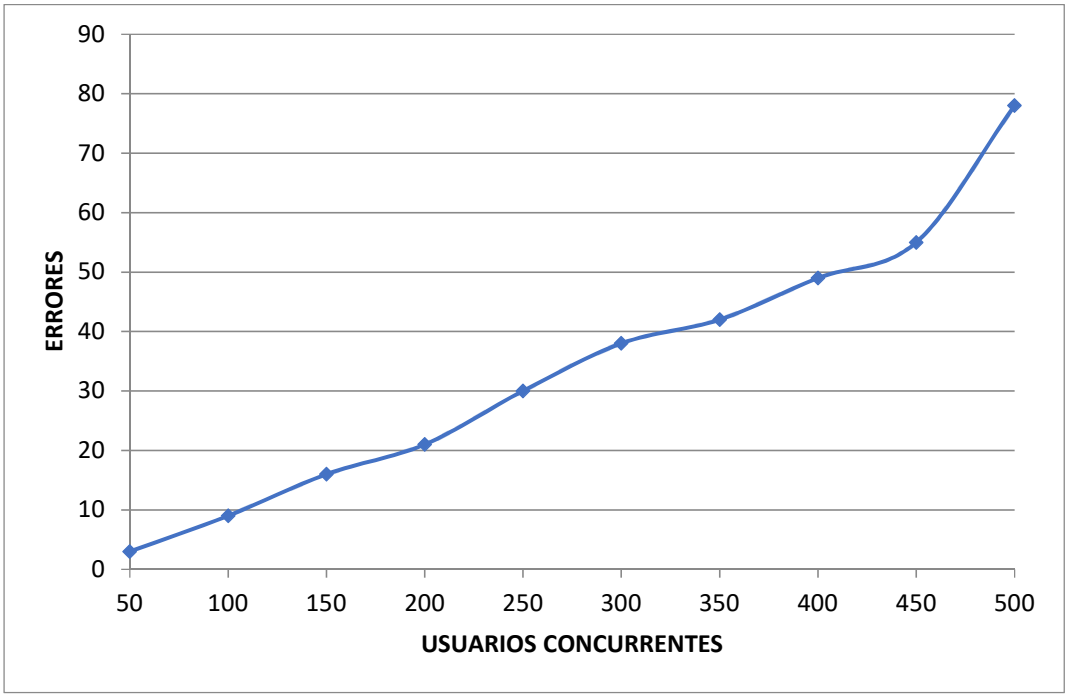

Fig. 29. Relación entre usuarios concurrentes y errores de usuario final

En la **Fig. 29** se describe el crecimiento de la aparición de errores de usuario final, a medida que crece la carga de usuarios concurrentes en el sistema; los valores representados no son puntuales ni absolutos, sino una aproximación basada en la media total de los errores reportados durante el periodo de pruebas.

Errores de usuario final desencadenados exclusivamente por las acciones del usuario que consume los servicios de la solución, es decir, aquellos que se conectan al entorno de red protegido por RADIUS. Estos problemas pueden variar desde ingresos inadecuados de datos en formularios de registro y autenticación, fallas de hardware en los equipos clientes o hasta violaciones a los protocolos de seguridad establecidos.

Del gráfico se puede inferir que existe una relación proporcional entre el número de usuarios conectados simultáneamente al sistema con los que se reportan por eventos de usuario final, por tanto, es posible afirmar que a mayor carga concurrente, el sistema será más propenso a sufrir colapsos relacionados con fallas humanas.

5.4.9. RELACIÓN CARGA DEL SISTEMA – ERRORES

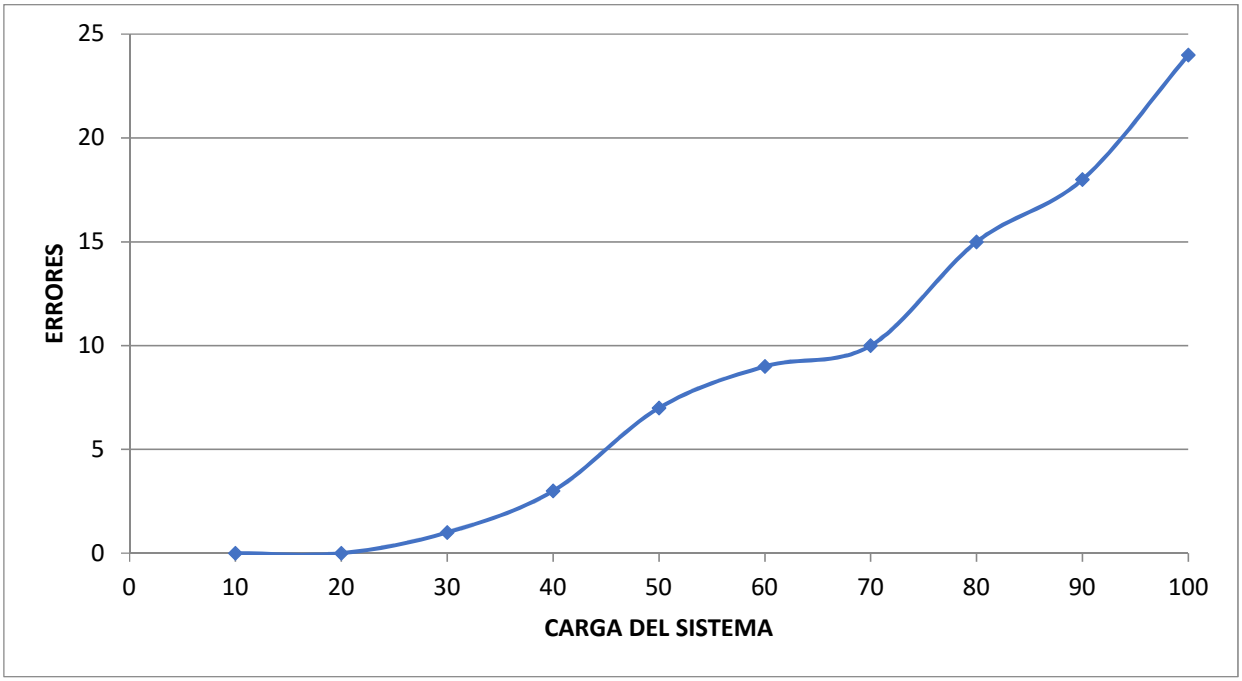

Fig. 30. Relación entre carga del sistema y errores.

En la **Fig. 30** se muestra el crecimiento de la aparición de errores, respecto al incremento de la carga del sistema respecto al uso del procesador y memoria. Los datos presentados son una ponderación de los diferentes factores de carga del sistema, contra una media aproximada de los casos de problemas reportados.

En esta gráfica se expresan los errores técnicos relacionados con la ejecución de procesos de software y rendimiento general de hardware. Tales errores pueden implicar desde colapsos por tareas incompletas hasta fallos críticos en el soporte físico de la solución.

Es posible identificar que, en cargas relativamente bajas, el sistema no sufre de problemas considerables. Sin embargo, durante picos de crecimiento en la carga, se presenta un crecimiento proporcional en el número de errores. De lo anterior, es posible deducir que el sistema alcanza su punto de equilibrio alrededor del 50% de la carga. Así mismo, se puede inferir, que aún en cargas extremas, no presenta un número significativo de errores.

6. CONCLUSIONES

El logro de poder construir un prototipo de un Access Point Wi-Fi, con el soporte de la arquitectura ARM en cuanto a procesador se refiere, un protocolo de autenticación y autorización para aplicaciones de acceso a la red o movilidad IP (FreeRADIUS Server) y su integración con el desarrollo de cuatro (4) posibles métodos de puntos de acceso, permite reducir los costos de adquisición, implementación, sencillez, movilidad y gestión para proyectos y soluciones que requieran similares características. Muy funcional para proveedores de Internet, sucursales empresariales y pequeños negocios con poca infraestructura tecnológica.

La diversidad en los diferentes métodos de acceso con características como la capacidad de despliegue de un formulario personalizable con el objeto de poder capturar datos del usuario, la capacidad de acreditar usuarios detrás de un sistema de autenticación digital descentralizado (OpenID), el acceso por medio de usuario y contraseña, y el no uso de credenciales, ayudan a reducir la inversión tecnológica, tanto en infraestructura como desarrollo de software de cualquier persona o compañía que requiera dos o más de estos servicios.

Usuarios potenciales del prototipo implementado son aquellos proveedores de servicios de Internet, cadena de restaurantes, centros comerciales, terminales de transporte aéreo y terrestre, clínicas/hospitales, centros educativos y cualquier compañía que tenga sucursales con poca infraestructura tecnológica. Siendo su principal uso el acceso al servicio de Internet y/o red corporativa, captura de datos, publicidad, seguridad en el proceso de autenticación y variedad en la fuente donde se alojan las credenciales de los usuarios (base de datos local, OpenID, LDAP, DA y ODBC).

El prototipo permite obtener diferentes beneficios que pueden ayudar a mejorar la seguridad en redes inalámbricas Wi-Fi y el buen uso de los recursos informáticos adquiridos a operadores de Internet. Esta característica se ve reflejada en el mismo momento que es reemplazado los procedimientos de protección como WEP, WPA-Personal o WPA2-Personal, por el uso de portales cautivos con métodos de autenticación soportados en 802.1X. La autenticación 802.1X usa un servidor de autenticación para validar las credenciales de los usuarios y proporcionar acceso a la red.

Otra de las bondades del proyecto realizado es la versatilidad de poder extender los servicios de una compañía que requiera capturar datos por medio de formularios web de forma fácil y sencilla. Para este caso, aplicarían clínicas (registro de salida de los pacientes), centros comerciales (captura de usuarios frecuentes para aplicación de servicios de Crossmedia), empresas (acceso invitados), terminales aéreas (ubicación de información de vuelos, servicios de acceso a la Internet, mapa del sitio, entre otros) y demás escenarios que requieran desplegar o capturar información de sus usuarios.

Gracias a los diferentes tipos de atributos soportados por FreeRADIUS, este servicio nos permite crear perfiles de clientes personalizados con características como: Control de ancho banda, VLAN (red de área local virtual), calidad de servicio, mensajes de bienvenida, entre otros. La diferenciación de servicio, ayuda a maximizar el buen uso de los recursos informáticos, la percepción del servicio por parte del usuario y el fortalecimiento de la relación cliente – empresa.

Para terminar, la pluralidad de operadores de Internet, la necesidad de usar los canales entregados por estos operadores de manera local, el requerimiento de despliegue de portales cautivos, la adaptación de los planes de datos a los usuarios del sistema (atributos FreeRADIUS), la difusión de contenidos publicitarios de manera personalizada por cada punto de acceso y la reducción en la cantidad de equipos electrónicos por sitio, obliga de alguna manera la creación de un dispositivo con las características del prototipo diseñado e implementado en este proyecto.

7. BIBLIOGRAFÍA

[1] H. X. H. Xia and J. Brustoloni, “Virtual prepaid tokens for Wi-Fi hotspot access,” *29th Annu. IEEE Int. Conf. Local Comput. Networks*, 2004.

[2] A. Stone, “For-Fee Hot Spots Strive to Make Wi-Fi Pay,” *Pervasive Comput. IEEE*, 2003.

[3] “20 Popular Wireless Hacking Tools,” 2016. [Online]. Available: http://resources.infosecinstitute.com/20-popular-wireless-hacking-tools-updated-for-2016/. [Accessed: 03-Apr-2016].

[4] X. Ding and J. Wei, “A scheme for confidentiality protection of OpenID authentication mechanism,” in *Proceedings - 2010 International Conference on Computational Intelligence and Security, CIS 2010*, 2010, pp. 310–314.

[5] “Seguridad avanzada en redes Wireless 802.1X.” [Online]. Available: http://www.jacksecurity.com/files/publications/Jack42.pdf. [Accessed: 03-Apr-2016].

[6] M. Rouse, “Captive Portal.” [Online]. Available: http://searchmobilecomputing.techtarget.com/definition/captive-portal. [Accessed: 14-Nov-2016].

[7] M. Rouse, “RADIUS (Remote Authentication Dial-In User Service),” 2007. [Online]. Available: http://searchsecurity.techtarget.com/definition/RADIUS.

[8] Internet Engineering Task Force (IETF), “RFC2865 -Remote Authentication Dial In User Service (RADIUS),” 2000.

[9] I. Cisco Systems, “How Does RADIUS Work ? Authentication and Authorization,” 2006.

[10] M. Rouse, “Authentication, Authorization, and Accounting (AAA).” [Online]. Available: http://searchsecurity.techtarget.com/definition/authentication-authorization-and-accounting. [Accessed: 14-Nov-2016].

[11] A. S. TANENBAUM and D. J. WETHERALL, *COMPUTER NETWORKS*, 5th ed. 2011.

[12] Internet Engineering Task Force (IETF), “RFC 7159. The JavaScript Object Notation (JSON) Data Interchange Format,” 2014.

ANEXO A

1. RADIUS-WEB API

El API público para integrar el sistema RADIUS con cualquier sistema informático externo, es un sistema simple de mensajes en formato JSON[12] que utiliza el protocolo HTTPS como capa de seguridad y método de conexión.

Este *API* no usa muchos nombres de funciones o métodos a través de la *URL*, como es habitual en estos casos; en su lugar todas las funciones son invocadas sobre la misma *URL* (que corresponde al *INDEX* del *WEB API*), distinguiendo cada posible acción por el contenido del paquete *JSON* que es recibido. Es preciso indicar que el *API* aquí mencionado cubre solo las funciones básicas del RADIUS, sin embargo, el protocolo es dinámico y extensible, de tal manera que se pueden agregar nuevas funcionalidades a petición del usuario.

No existe una única *URL* global mediante la cual se pueda acceder al *API*, ya que cada servidor RADIUS está sometido a diferentes configuraciones de red que por lo general redirigen o enmascaran tanto los nombres de dominio como la dirección IP o incluso el puerto de servicio. Sin embargo, en todos los casos se sigue el siguiente esquema:

https://[IP or HOST]:PORT/rsa/vendor/wa/

El puerto por defecto es el 9701, sin embargo, esto puede cambiar en virtud de la configuración de red y el método de publicación que se use, especialmente en casos donde el acceso es gestionado por un *Firewall*. Es importante mantener siempre el *slash (/)* al final de la *URL* ya que esto indica el punto de inicio de la aplicación, concretamente indica la invocación al *INDEX*.

Para usar el *API* no es necesario definir cabeceras (*HEADERS*) especiales, a parte de las necesarias para establecer una petición *HTTPS*. Dicho esto, existen solo dos condiciones a cumplir obligatoriamente:

1. Todas las peticiones deben usar el método *POST (método POST envía los datos de forma que no se pueden ver [en un segundo plano u "ocultos" al usuario])*. Cualquier petición por otro método (*GET, PUT, DELETE, etc*) será rechazada como un error de petición.
2. Todas las peticiones deben definir la cabecera "*Content-Type*" como "*application/json*": Si tal cabecera no está presente o contiene un valor diferente, la petición será rechazada con un error "*Unsupported Media Type*".

Cualquier otra cabecera es innecesaria y se aconseja no agregar datos inútiles en las peticiones.

Se debe tener en cuenta que *API* es un servicio *WEB* que puede ser habilitado o deshabilitado mediante el panel de administración de servicios en *WEBMIN* de RADIUS y para poder realizar peticiones al *API*, es necesario contar con un *CLIENT_ID* el cual se crea desde el menú SERVICIOS – *Web API* dentro del *WEBMIN* de RADIUS. Necesitará privilegios de administrador para realizar esta acción.

Para crear el *CLIENT_ID* se debe proporcionar un nombre de identificación y seleccionar que acciones quiere habitar para ese cliente en concreto; es posible crear tantos clientes como

requiera y administrar individualmente sus privilegios para acceder a determinadas acciones, de tal manera que puede crear clientes para funciones específicas, por ejemplo, un cliente con acceso únicamente a funciones de consulta.

Una vez creado el *CLIENT_ID* obtiene los siguientes datos:

1. *CLIENT_ID*: Es el identificador único de cada cliente que lo autentica y autoriza sobre el sistema RADIUS. Concretamente se trata de una cadena hexadecimal de 128 bytes. Por ejemplo:

CLIENT_ID:91F8C5A62169536B7639E34CAFC70E62282FEE6A988FFC3A387D7F5E9E00D795E1713224A567122BAE10950B977E1413E199DB658439766920372207983B66E8.

1. *OUT_WRAP*: Es una clave de cifrado simétrico con la cual deberá procesar todas la *KEYS* que envié como peticiones al *API*. Concretamente se trata de una cadena de 16 bytes (128 bits) en formato Base64. Por ejemplo: OUT_WRAP: Q29uY3JldGFtZW50.

2. *IN_WRAP:* Es una clave de cifrado simétrico con la cual RADIUS procesará todas la *KEYS* que envié como respuestas mediante el API. Concretamente se trata de una cadena 128 bytes en formato Base64, idéntica al OUT_WRAP.

3. *DEAD_POOL*: Es un conjunto de 64 números primos aleatorios de 4 dígitos, los cuales se usan como *SALT* para crear cada *KEY* de petición y respuesta, tanto su aplicación cliente como el servidor RADIUS usaran el mismo conjunto de números para esta labor.

   Por ejemplo:

   1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087,2711, 2713, 2719, 2729, 2731, 2741, 2749, 2753, 3163, 3167, 3169, 3181, 3187, 3191, 3203,3209,4637, 4639, 4643, 4649, 4651, 4657, 4663, 4673, 5171, 5179, 5189, 5197, 5209, 5227, 5231, 5233, 7127, 7129, 7151, 7159, 7177, 7187, 7193, 7207, 8087, 8089, 8093, 8101, 8111, 8117, 8123, 8147, 9137, 9151, 9157, 9161, 9173, 9181, 9187, 9199.

5. *REQUEST_KEY* y *RESPONSE_KEY* son atributos necesarios en cada petición (*request*) que se envié al *API* y están presentes en cada respuesta (*response*) que devuelve el *API*. Son el único mecanismo que permite verificar que la aplicación cliente efectivamente tiene privilegios para interactuar con el servidor RADIUS, las respuestas obtenidas son efectivamente enviadas por el servidor RADIUS consultado.

2. FORMATO GENERAL DE LOS PAQUETES JSON

A continuación, se muestra el formato general que deben seguir todos los paquetes JSON tanto de peticiones como de respuestas:

```
{
        “action”:”{ACTION}”,
        “key”:”RESQUEST_KEY | RESPONSE_KEY”,
        “atr1”:”val1”,
        …
        “atrN”:”valN”,
        [”status”:”CODE”,]
        [”desc”:”INFO”,]
}
```

El atributo *action* es siempre obligatorio e indica la acción que se solicita o a la que se responde.

El atributo *key* es siempre obligatorio e indica la *KEY* de seguridad que se usa para ese paquete, debe generarse según el esquema mencionado anteriormente.

Los atributos a *atr1-N* y sus *val1-N* representan todos los campos que se envían para tal acción, varían según el caso.

Los *val1-N* pueden ser tanto valores simples, como arreglos u objetos *JSON* anidados.

Los atributos *status* y *desc* están presentes solo en los paquetes JSON que corresponden a las respuestas enviadas por el servidor, en caso de que el status sea 0, es decir exitoso, no se envía el atributo *desc*, ya que este se usa para indicar el detalle de los errores que han ocurrido.

A continuación, se detallan las acciones disponibles con sus respectivos esquemas de petición y respuesta.

2.1. LIST-GROUPS

Esta acción permite obtener la lista de grupos completa que se encuentra en el servidor.

Request:

```
{
        “action”:”LIST-GROUPS”,
        “key”:”RESQUEST_KEY”,
}
```

Response:

```
{
```

“action”:”LIST-GROUPS”,
“key”:”RESPONSE_KEY”,
“groups”:[”group1”, “group2”, ...]
[”status”:”CODE”,]
[”desc”:”INFO”,]
}

2.2. LIST-USERS

Esta acción permite obtener la lista de usuarios completa que se encuentra en el servidor.

Request:

{
“action”:”LIST-USERS”,
“key”:”RESQUEST_KEY”,
}

Response:

{
“action”:”LIST-USERS”,
“key”:”RESPONSE_KEY”,
“users”:[”username1”, “username2”, ...]
[”status”:”CODE”,]

2.3. USERS-FROM-GROUP

Esta acción permite obtener la lista de usuarios de un grupo concreto.

Request:

{
“action”:”USERS-FROM-GROUP”,
“key”:”RESQUEST_KEY”,
“group”:”groupname”,
}

Response:

{
“action”:”USERS-FROM-GROUP”,
“key”:”RESPONSE_KEY”,
“users”:[”username1”, “username2”, ...]
[”status”:”CODE”,]

[”desc”:”INFO”,]
}

2.4. GROUPS-FROM-USER

Esta acción permite obtener la lista de grupos a la que pertenece un usuario concreto.

Request:

{
“action”:”GROUPS-FROM-USER”,
“key”:”RESQUEST_KEY”,
“user”:”username”,
}

Response:

{
“action”:”GROUPS-FROM-USER”,
“key”:”RESPONSE_KEY”,
“groups”:[
{
“group”: “groupname”,
“priority”: “prioritylevel”
}, ...]

[”status”:”CODE”,]
[”desc”:”INFO”,]
}

2.5. ATT-FROM-GROUP

Esta acción permite obtener la lista de atributos asignados a un grupo.

Request:

{
“action”:”ATT-FROM-GROUP”,
“key”:”RESQUEST_KEY”,
“group”:”groupname”,
}

Response:

{

“action”:”ATT-FROM-GROUP”,
“key”:”RESPONSE_KEY”,
“attributes”:[{”name”: “attname”, “op”:”operator”, “value”: “attvalue”}, …]
[”status”:”CODE”,]
[”desc”:”INFO”,]
}

2.6. ATT-FROM-USER

Esta acción permite obtener la lista de atributos asignados a un usuario. Tanto aquellos que se le han asignado directamente, como aquellos que recibe de sus respectivos grupos.

Request:

{
“action”:”ATT-FROM-USER”,
“key”:”RESQUEST_KEY”,
“user”:”username”,
}

Response:

{
“action”:”ATT-FROM-USER”,
“key”:”RESPONSE_KEY”,
“attributes”:[
{
“group”: “groupname”,
“priority”: “prioritylevel”,
“attributes”: [
{
“name”: “attname”,
“op”:”operator”,
“value”: “attvalue”
}, …]
}, …]
[”status”:”CODE”,]
[”desc”:”INFO”,]
}

2.7. SET-ATT-TO-GROUP

Esta acción permite fijar un atributo a un grupo concreto.

Request:

```
{
        “action”:”SET-ATT-TO-GROUP”,
        “key”:”RESQUEST_KEY”,
        “group”:”groupname”,
        “att”:”attributename”,
        “op”:”operator”,
        “value”:”attvalue”,
        [”order”:”orderforatt”],
}
```

Response:

```
{
        “action”:”SET-ATT-TO-GROUP”,
        “key”:”RESPONSE_KEY”,
        [”status”:”CODE”,]
        [”desc”:”INFO”,]
}
```

Si se envía un atributo con operador “*NULL*” y valor “*NULL*” el sistema busca ese atributo y lo elimina. El campo opcional *order* le indica al sistema que debe fijar ese atributo en esa posición específica: si el valor de *order* es superior al total de los atributos asignados, el atributo se fija en último lugar. Si el valor de *order* es 0, el atributo se fija en primer lugar. Si el valor de order es negativo, se fija en orden inverso del final al inicio. Por último, si no envía *order,* el atributo se fija siempre al final.

2.8. SET-ATT-TO-USER

Esta acción permite fijar un atributo a un usuario concreto.

Request:

```
{
        “action”:”SET-ATT-TO-USER”,
        “key”:”RESQUEST_KEY”,
        “user”:”username”,
        “att”:”attributename”,
        “op”:”operator”,
        “value”:”attvalue”,
        [”order”:”orderforatt”],
```

}

Response:

{
“action”:”SET-ATT-TO-GROUP”,
“key”:”RESPONSE_KEY”,
[”status”:”CODE”,]
[”desc”:”INFO”,]
}

Si se envía un atributo con operador “*NULL*” y valor “*NULL*” el sistema buscara ese atributo y lo eliminara. EL campo opcional *order* se trata igual que para la petición *SET-ATT-TO-GROUP*.

2.9. COM-ATT-FROM-USER

Esta acción permite obtener la lista de atributos asignados y activos para un usuario, es decir aquellos que el sistema procesa para el usuario en ese momento, sin indicar si proceden de un grupo o del mismo usuario. Es equivalente al RADIUS RESPONSE que se genera internamente al preguntar por un usuario.

Request:

{
“action”:”COM-ATT-FROM-USER”,
“key”:”RESQUEST_KEY”,
“user”:”username”,
}

Response:

{
“action”:”COM-ATT-FROM-USER”,
“key”:”RESPONSE_KEY”,
“attributes”:[
{
“name”: “attname”,
“op”:”operator”,
“value”: “attvalue”
}, ...]
[”status”:”CODE”,]
[”desc”:”INFO”,]
}

2.10. META-ATT-FROM-GROUP

Esta acción permite obtener la lista de meta atributos de un grupo, esta información varía en virtud de la configuración de seguimiento que se tenga en el sistema; sin embargo, estos datos suelen contener como mínimo los datos de sesión del grupo, tales como consumos medios de ancho de banda, duración media de sesión, media de usuarios conectados, direcciones físicas y lógicas de los usuarios, etc.

Request:

```
{
        "action":"META-ATT-FROM-GROUP",
        "key":"RESQUEST_KEY",
        "user":"username",
}
```

Response:

```
{
        "action":"META-ATT-FROM-GROUP",
        "key":"RESPONSE_KEY",
        "metas":[
                {
                        "name": "metaname",
                        "val1":"metaval1",
                                ...
        "valN": "metavalN"
                }, ... ]
        ["status":"CODE",]
          ["desc":"INFO",]
}
```

2.11. META-ATT-FROM-USER

Esta acción permite obtener la lista de meta atributos de un usuario, esta información cambia en virtud de la configuración de seguimiento que se tenga en el sistema; sin embargo, estos datos suelen contener como mínimo los datos de sesión del usuario, tales como consumos de ancho de banda, duración de sesión, direcciones físicas y lógicas, etc.

Request:

{

“action”:”META-ATT-FROM-USER”,
“key”:”RESQUEST_KEY”,
“user”:”username”,

}

Response:

{

“action”:”META-ATT-FROM-USER”,
“key”:”RESPONSE_KEY”,
“metas”:[
{
“name”: “metaname”,
“val1”:”metaval1”,
…
“valN”: “metavalN”
}, …]
[”status”:”CODE”,]
[”desc”:”INFO”,]

}

ANEXO B

WORKERS

Definición.

Bajo el modelo de diseño sustractivo, es posible definir un "worker" como un componente de software que posee las siguientes propiedades:

1. Ser funcional. Esto quiere decir que su ejecución es independiente de los datos con que trabaja, permitiendo con ello ser fácilmente reutilizado por diferentes procesos.

2. No tener estado. Esto implica que su ejecución es única, directa y espontanea; por tanto, un worker no puede usarse como servicio, pero sí que puede ser invocado en múltiples ocasiones por múltiples servicios, incluso en paralelo, sin que sus segmentos de memoria se vean mezclados.

Nivel.

Dada su naturaleza funcional, un worker puede considerarse de diferentes niveles en virtud del papel que desempeña dentro de un sistema concreto; así, un worker que es invocado directamente por un servicio o aplicación, se considera de nivel uno o primario, pues solo existe una llamada en la pila de procesos que lo invoca. En caso de que un worker sea invocado por otro worker este se considerará en virtud del nivel del worker que lo invocó.

Los workers de nivel uno o primarios son por lo general construcciones conceptuales que definen la funcionalidad general de worker, y que invocan workers de segundo nivel para realizar labores más concretas, de esta forma un worker primario es una unidad más abstracta que se encarga de procesos generales, mientras que los workers secundarios y subsiguientes tienden a ser más puntuales, realizando pocas tareas con resultados más inmediatos, generando así una arquitectura de delegación y un resultado final en propagación inversa.

Workers primarios.

A continuación, se describen los workers primarios que permiten la correcta ejecución del ecosistema RADIUS.

PCW – Worker de configuración primaria.

Es el corazón de todo el sistema de gestión administrativa del ecosistema RADIUS, este componente se encarga de actualizar los ficheros de configuración de todos los servicios y aplicaciones de la solución. Realiza su trabajo en base a dos componentes:

1. La tabla de variables de la base de datos donde se almacenan todas las variables de configuración como registros independientes.

2. Una plantilla de formato. Ya que cada servicio o aplicación usa diferentes formatos para almacenar sus configuraciones, es necesario definir una plantilla inicial bajo la cual se puedan construir los ficheros de configuración. Algunos servicios como por ejemplo el DNS, hacen uso de múltiples archivos con diferentes formatos, sin embargo, esto es transparente para el PCW ya que este trabaja solo con un fichero de configuración a la vez.

Invocación:

Para invocar a PCW es preciso ejecutar la siguiente orden con privilegios administrativos:

/opt/it/workers/pcw [target] [template]

Donde [target] es el fichero de configuración a escribir, por ejemplo /etc/network/interfaces y [template] es la ruta absoluta al fichero que contiene la plantilla de formato.

Resultado:

PCW devuelve un único código de respuesta, a saber: 0 si ha realizado su labor correctamente o 1 en caso de presentar un error. Como resultado tangible de su trabajo se obtiene el fichero de configuración actualizado.

PCW reemplaza directamente el contenido del fichero objetivo, por lo que, si se desea mantener una copia de la versión anterior de dicho fichero, se debe realizar este proceso antes de invocar a PCW.

Código PCW:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
#   PCW
#

from it.metas.targets import TargetConfig as tg
from it.metas.targets import DefineTargets as DT
from it.tools.sql import Engine as sql
from it.tools.file import FileWorker as fw
from it.tools.file import FixEncode as fe

def getTargetData(tar):

    try:

        querys = tg.getSQL(tar)
        data = sql.exe(querys)
        return date.parseJSON()

    except:

        return 1
```

```
def getTemplateFormat(tem):

    try:

        f = open(tem, 'r')
        fd = f.read()
        f.close()

        typef = FW.getType(fd)
        if(typef in TD):

            container = FW.parseTemplate(fd, TD[typef])
            return container.inDicMode()

        else:

            return 1

    except:

        return

def applyData2Template(dat, tem):

    try:

        dataFix = fe.fixEncode(dat, 'utf-8')
        finalText = fw.processTemplate(tem, dataFix)

        return finalText

    except:

        reurn 1

def main(args):

    tarData = getTargetData(args[1])
    if(tarData != 1):

        temData = getTemplateFormat(args[2])

        if(temData != 1):

            text = applyData2Template(tarData, temData)

            if(text != 1):

                try:

                    f = open(args[0], 'w')
                    f.write(text)
                    f.close()

                    return 0

                except:
```

```
                    return 1

    return 1

if __name__ == '__main__':
    import sys
    sys.exit(main(sys.argv))
```

RCW – Worker de ejecución en arranque

Se encarga de gestionar el orden de arranque de los servicios, proporcionar configuraciones iniciales y lanzar aplicaciones de soporte, todo esto durante el inicio del sistema. RCW es un worker de configuración, es decir no es una aplicación como tal sino un perfil de pseudo servicio el cual se configura sobre el demonio maestro SystemD para habilitar el script de ejecución ubicado en /etc/rc.local.

Invocación:

RCW es invocado automáticamente por el demonio maestro SystemD durante el inicio del sistema, no recibe parámetros ni consume fuentes directamente, salvo lo consignado en el script rc.local.

Resultado:

RCW genera resultados diversos en virtud de que se encuentre configurado en rc.local, estos resultados pueden ser consultados a través del Log de SystemD o por el comando journalctl.

Código RCW:

```
#
#       IT Radius - RCW
#

[Unit]
Description=/etc/rc.local Compatibility
ConditionPathExists=/etc/rc.local

[Service]
Type=forking
ExecStart=/etc/rc.local start
TimeoutSec=0
StandardOutput=tty
RemainAfterExit=yes
SysVStartPriority=99

[Install]
WantedBy=multi-user.target
```

CTW – Worker de ejecución cronológica

Este worker se encarga de agendar las tareas que deben realizarse periódicamente, permite por tanto registrar procesos en la tabla de control del demonio CRON, fijando tanto el momento como el nivel de privilegios con que debe ejecutarse un proceso. Así mismo CTW extiende las funcionalidades de CRON permitiendo fijar fechas concretas, cantidad de veces que debe ejecutarse un proceso y validaciones lógicas adicionales que deben completarse antes de ejecutar un proceso.

CTW depende de un fichero de configuración JSON que contiene todas las especificaciones de los procesos que debe gestionar; es posible desplegar múltiples instancias de CTW con diferente configuraciones fuentes.

Invocación:

CTW es invocado automáticamente cada segundo por el demonio CRON, mediante el intérprete de Python3 y con privilegios de administrador. En el fichero *crontab* se define cuantas instancias de CTW se ejecutarán y que ficheros fuente se usarán para cada una. Esta configuración se realiza mediante PCW.

Resultado:

Por lo general CTW no genera resultados, salvo que se presenten fallas en alguno de los procesos que se le han registrado, en tal caso su salida puede verse en el fichero /var/log/ctw.log

Código CTW:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
#   CTW
#

import os
import commands
import time

from it.metas.targets import DefineTargets as DT
from it.metas.apps import ProcessWorker as pw
from it.tools.file import FileLoader as fl
from it.tools.file import logger as log
from it.tools.executor import ExecWorker as ew
from it.tools.users import UserWorker as uw
from it.tools import calendar as cal


def main(args):

    try:
```

```
        jpo = fl.getDictFromJSON(args[1])

        for key in jpo.keys():

            pro = jpo[key]
            meta = pw.parceProcess(pro)

            if(DT[meta.top] == 'OS'):
                engine = os
            else:
                engine = commands

            machine = ew.create(engine)
            user = uw.getUser(meta.user)
            now = cal.getNow(time.time())

            machine.do(meta.orders, user, now)

    except Exception as ex:

        log.put("ctw", ex)

    return 0

if __name__ == '__main__':
    import sys
    sys.exit(main(sys.argv))
```

WRW – Worker de peticiones Web

Es el motor maestro encargado de emitir las peticiones que el sistema RADIUS hace hacia proveedores en la WEB, soporta los métodos GET y POST, así como la posibilidad de construir cuerpos de peticiones en diferentes formatos, como XML o JSON.

Invocación:

Para invocar a WRW es preciso ejecutar la siguiente orden:

/opt/it/workers/wrw [desc]

Donde [desc] es un fichero descriptor con los datos necesarios para realizar la petición al servidor remoto. El fichero descriptor se construye con la ayuda de PCW.

Resultado:

RWR devuelve la respuesta obtenida del servidor en forma de texto plano, con estructura JSON; en caso de que el servidor remoto devuelva contenido en formato binario directo, como imágenes, audio o video, este contenido se devolverá codificado en base64. El resultado se devuelve directamente por salida estándar, a menos que se invoque a RWR con una re-dirección de salida o una tubería.

Código WRW:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
#   WRW
#

from it.tools.file import FileLoader as fl
from it.tools.file import FileChecker as fc
from it.tools.file import FixEncode as fe
from it.net.request import Request as rq
from it.net.response import Response as rs

def main(args):

    try:

        mode = fc.getTypeOf(args[1])
        rdata = fl.loadRequestData(args[1], mode)

        rq.send(rdata)

        res = rs.parseToJSON(rq.recvData)

        return fe.fixEncode(res, 'utf-8')

    except:

        return -1

    return 0

if __name__ == '__main__':
    import sys
    sys.exit(main(sys.argv))
```

GSW – Worker de sincronización general

Este worker permite sincronizar las credenciales de usuarios de proveedores externos en las tablas de cache de la base de datos local, permitiendo así mantener una base de respuesta para peticiones en caso que el proveedor no se encuentre disponible. GSW realiza peticiones de control a los proveedores externos, para obtener las credenciales a almacenar; tales peticiones de control pueden ir desde sencillas consultas SQL en caso de que el proveedor externo sea un RDMS, hasta complejas rutinas de consultas NTL, si se trata de un controlador de directorio activo.

GSW requiere de configuración previa, mediante perfiles que definen a cada proveedor y que contienen la serie de procesos que deben realizarse para el proceso de sincronización.

Invocación:

Para invocar a GSW es preciso ejecutar la siguiente orden con privilegios de administrador:

/opt/it/workers/gsw [profile]

Donde [profile] es un fichero que define el perfil del proveedor a sincronizar.

Resultado:

Los resultados de GSW se pueden observar al consultar las tablas de cache de credenciales, las cuales se borran y actualizan al ejecutar este worker. Los errores que se puedan presentar se almacenan en el fichero /var/log/gsw.log.

Código GSW:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
#   GSW
#

import commands

from it.metas.apps import ProcessWorker as pw
from it.tools.sql import Engine as sql
from it.tools.file import FileLoader as fl
from it.tools.file import logger as log
from it.tools.executor import ExecWorker as ew
from it.tools.users import UserWorker as uw
from it.tools.exceptions import ConfigExecption as ce

def main(args):

    try:

        prof = fl.getProviderProfileFromJSON(args[1])

        if(prof.tp == 'ADC'):
            from it.providers import ADC as pe
        elif(prof.tp == 'LDAP'):
            from it.providers import LDAP as pe
        elif(prof.tp == 'ODBC'):
            from it.providers import ODBC as pe
        else:
            ce.generateProfileEx(prof)

        meta = pw.parceProcess(prof.processList)
        machine = ew.create(commands)
        user = uw.getUser('root:root')

        pe.createSyncJob()
        pe.user = user
        pe.runner = machine.getRuntimeInBg()
        pe.jobs = meta.getProcessList()

        if(pe.verify()):
```

```
            querys = []

            for q in pe.doJob():
                querys.append(q)

            sql.exe(querys)

        else:
            ce.generateFailureEx(pe.fails)

    except Exception as ex:

        log.put("gsw", ex)

    return 0

if __name__ == '__main__':
    import sys
    sys.exit(main(sys.argv))
```