
# Arquitectura para automatizar respuesta a incidentes de seguridad de la información relacionados con ataques internos mediante la ejecución de técnicas spoofing

**Cesar Augusto Rios Agudelo**

Instituto Tecnológico Metropolitano

Facultad

Ciudad, Colombia

2020

# Arquitectura para automatizar respuesta a incidentes de seguridad de la información relacionados con ataques internos mediante la ejecución de técnicas spoofing

**Cesar Augusto Rios Agudelo**

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de:

**Magister en Seguridad informática**

Director (a):

M.Sc. Milton Javier Mateus Hernandez.

Línea de Investigación:

Ciencias Computacionales.

Instituto Tecnológico Metropolitano

Facultad

Ciudad, Colombia

2020

*A Dios por haberme regalado la vida*
*A mis padres por sus esfuerzos que hicieron de mí una persona de bien.*
*A mi pareja por apoyarme y hacerme sentir que puedo lograr cualquier meta que me proponga.*
*A mi familia por siempre permanecer unida.*
*A mis docentes por cada enseñanza entregada.*

# Agradecimientos

Quiero agradecer de manera muy especial a Milton Javier Mateus mi asesor técnico y metodológico que con su apoyo, observaciones, recomendaciones y tiempo entregado fue posible desarrollar y entregar este trabajo de investigación, a Hector Vargas quien desde el comienzo de este proceso estuvo especialmente dispuesto a facilitar todo lo relacionado al desarrollo de esta tesis, quien por sus esfuerzos y buena gestión ha ido posicionando esta gran Maestría en la región en la Institución Universitaria ITM y finalmente quiero agradecer a todo el grupo de docentes quienes con sus enseñanzas y esfuerzos permitieron guiar el cumplimiento de este gran logro.

# Resumen

En esta tesis de maestría se examina y desarrolla una arquitectura basada en un sistema central que permite minimizar los riesgos asociados a los incidentes de seguridad que aprovechan las debilidades existentes en los esquemas de comunicación, estas debilidades facultan la ejecución de ataques spoofing que generalmente son ejecutados por personas mal intencionadas al interior de las organizaciones, también conocidos como *malicious insider threat*. Para alcanzar este propósito se define una metodología de atención de incidentes que en primera instancia protege la red mediante el endurecimiento automático de los dispositivos, seguidamente apoya la atención de incidentes mediante estrategias de detección y análisis de eventos tipo spoofing y finalmente de procedimientos automáticos que permitan aislar la amenaza encontrada y notificar al administrador para que pueda aplicar las respectivas acciones disciplinarias sobre el agente detectado.

**Palabras clave:**

**Spoofing:** Técnicas que permiten realizar suplantación de identidades mediante la falsificación de los datos en un esquema de comunicación.
**Malicious insider threat:** Personas dentro de una organización que ejecutan acciones que afectan la seguridad de los procesos y/o sistemas, generando un impacto negativo para la empresa.
**Atención de incidentes de seguridad de la información:** Metodología que permite responder de manera ágil y eficiente a incidentes que atenten contra la seguridad de la información de una compañía.
**Endurecimiento:** Técnicas de aseguramiento que permiten minimizar el grado de exposición de vulnerabilidades de un sistema informático.
**Vulnerabilidad:** Debilidad en un sistema que permite la ocurrencia de un evento de seguridad.

# Abstract

This master's thesis examines and develops an architecture based on a central system that allows minimizing the risks associated with security incidents that take advantage of existing weaknesses in communication schemes. These weaknesses enable the execution of spoofing attacks that are usually executed by malicious people inside organizations, also known as malicious insider threat. To achieve this purpose, an incident care methodology is defined that firstly protects the network by automatically hardening the devices, then supports the incident attention by means of detection strategies and analysis of spoofing events and finally automatic procedures that allow Isolate the threat encountered and notify the administrator so that he can apply the respective disciplinary actions on the detected agent.

**Keywords:**

**Spoofing:** Techniques that allow to perform the impersonation by falsifying the data in a communication scheme.
**Malicious threat:** People within an organization who execute actions that affect the security of processes and / or systems, generating a negative impact for the company.
Attention of information Security Incident: A methodology that allows to respond in an agile and efficient way to incidents that threaten the security of the information of a company.
**Hardening:** Assurance techniques that allow to minimize the degree of exposure to the vulnerability of a computer system.
**Vulnerability:** Weakness in a system that allows the appearance of a security event.

# Contenido

# Lista de figuras

# Lista de tablas

**Pág.**

# Introducción

Las organizaciones día a día se enfrentan a miles de amenazas cibernéticas externas que en caso de materializarse podrían poner en riesgo los objetivos de la misma, es por ello que muchas de ellas invierten grandes cantidades de dinero con fin de fortalecer su defensa perimetral, sin embargo, en ocasiones personas mal intencionadas al interior de las empresas pueden causar un impacto negativo mucho más alto que las propias amenazas externas, esto dado a que dichas personas generalmente logran obtener un conocimiento importante relacionado con los procesos y esquemas de seguridad implementados en la organización, además de contar con accesos y privilegios sensibles a cierto tipo de información. El 6 de febrero del 2020 la reconocida empresa Panda Security desvela cifras alarmantes del Ponemon Institute en donde indican un aumento en los últimos años en la frecuencia y costo de los incidentes de seguridad internos equivalente al 47%. otra cifra muy impactante es el hecho de que el 60% de las organizaciones sufrieron más de 30 incidentes por atacantes internos al año y entre las organizaciones encuestadas, hubo 4.716 incidentes causados por estas amenazas internas. [1]

Es importante mencionar que dentro de los atacantes internos o también conocidos como malicious insiders o insiders threats se encuentran tres clases de grupos; los realizados por personas negligentes de manera accidental, las personas abusivas que aprovechan sus privilegios de acceso a información sensible de la organización y los maliciosos que son personas que están motivadas por ganancias financieras, espionaje o venganza. [2], adicional, el Ponemon institute revela datos importantes en donde cuantifica los costos y frecuencia de cada uno de estos ataques, dejando el grupo de los insider negligentes como

la amenaza más frecuente con un 62% pero con el costo más bajo por incidente reportado $307.000 US, seguido de los insider abusivos con una frecuencia del 23% y con el costo más alto por incidente reportado $871.000 US y finalmente están los insider maliciosos con una frecuencia del 14% con un costo no insignificante por incidente de $ 756.000 US, dejando así un costo anual de $ 11,45M US en perdidas asociadas a estas amenazas internas [3].

Las estadísticas previamente citadas dejan bastante preocupación pues las personas internas pueden facultar incidentes de seguridad con un impacto bastante alto en las organizaciones, sobre todo, cuando estos ataques son facultados por personas con intenciones maliciosas o con accesos privilegiados a información o esquemas de infraestructura tecnológica de la empresa y es aún más preocupante si tenemos en cuenta las debilidades actuales en los esquemas de comunicación IPV4 usados por la mayoría de organizaciones en su red privada, estos modelos son los mismos en los que se basaban tiempo atrás las comunicaciones y si es cierto que han tenido pequeñas modificaciones aún no logran mitigar los fallos de seguridad que han traído desde su creación, pues es claro que en antaño los modelos de comunicación fueron creados con la única necesidad de poder interconectar los diferentes sistemas y no había lugar para pensamientos maliciosos. Estos problemas salieron a luz años después y contrario a los modelos teóricos de comunicación las herramientas y técnicas que aprovechan estas vulnerabilidades si han ido evolucionado permitiendo obtener una eficacia y efectividad mayor sobre los ataques orientados a aprovechar estas debilidades [4]. Uno de los casos más críticos por su simpleza se evidencia en las técnicas de suplantación e intercepción de tráfico tipo spoofing donde existen numerosas técnicas de ataque que automatizan procedimientos cuya finalidad radica en la interceptación, robo y/o modificación de la información que viaja sobre la red, el caso se agrava aún más cuando estas técnicas de interceptación de tráfico y suplantación son de fácil ejecución en las cuales existen miles de referencias que guían a la persona mal intencionada en un paso a paso su fabricación.

Y si a esto sumamos las prácticas y tendencias actuales en la digitalización de los productos presentes en la revolución 4.0 [5] mediante la adopción de tecnologías que permiten maximizar y optimizar la eficacia de los servicios por medio del análisis de datos en tiempo real y la integración de servicios vemos que las organizaciones incorporan una dependencia aún más de la seguridad, las comunicaciones y la tecnología, es tanto,

incluso las organizaciones más industrializadas y enfocadas únicamente en tecnologías de operación (TO) han venido adoptando tecnologías convergentes MES (Manufacturing Execution System) del mundo de TI, pues estas redes de operación guardan información muy valiosa en sus sensores, actuadores, PLC y demás, que al ser analizada por tecnologías BI (Business Intelligence) logran optimizar en gran medida los procesos operativos llevados por estas redes [6].

Un elemento adicional y presente sobre estas industrias es el hecho de que los administradores de las interfaces humanas (HMI) encargados de supervisar la red de operación y en ocasiones escribir instrucciones en los PLC o RTU, generalmente se encuentran conectados a una red de TI bajo protocolos IPv4, volviendo la red OT vulnerable a todas las deudas técnicas que traen los protocolos de comunicación IPv4 desde su diseño [7]

Sin embargo, existen metodologías y sistemas de gestión de seguridad de la información que permiten detectar y dar respuesta de manera oportuna a este tipo de incidentes, no obstante, esta atención requiere de metodologías y procedimientos eficaces que desafortunadamente muy pocas organizaciones disponen evidenciando así una falta de capacidad para identificar y controlar una filtración en los primeros minutos, esto se debe precisamente por la cantidad de recursos en TI necesarios para las fases de preparación, detección, contención y erradicación [8]. Generalmente estos recursos requieren de herramientas tecnológicas que permitan preparar a la organización para afrontar este tipo de incidentes, de mecanismos de detección que permitan identificar y analizar la amenaza y de recursos humanos con conocimientos técnicos importantes que ejecuten acciones de contención y erradicación tras la alerta o notificación del incidente., es por ello que está tesis de maestría tiene como propósito proporcionar una arquitectura que permita automatizar la atención de incidentes relacionados con amenazas spoofing, permitiendo de manera automática aplicar políticas en el aseguramiento en los dispositivos de red de manera que no sea tan sencilla la ejecución de este tipo de ataques, de mecanismos de detección y análisis que permitan identificar y valorar oportunamente los ataques que hayan trascendido los controles de preparación, de mecanismos y procedimientos de mitigación y contención que permita aislar la amenaza de manera automática y finalmente de esquemas de notificación que ayuden al administrador a la toma de medidas disciplinarias.

En este orden de ideas la presente tesis de maestría entrega tres importantes aportes o contribuciones sobre los cuales se enmarca la metodología y desarrollo de la investigación, sobre una primera instancia se tiene un producto final el cual permite de manera automática aplicar mejores prácticas de implementación de infraestructura en dispositivos de red el cual permitirá preparar la red para ataques spoofing, esto sin que las configuraciones aplicadas llegasen a afectar la disponibilidad de la red que se quiere fortalecer, en segunda medida un esquema de detección que permita identificar de manera rápida y oportuna ataques internos relacionados con técnicas de ARP Spoofing, DNS Spoofing y DHCP Spoofing, finalmente una arquitectura de interacción con la red de datos la cual permita aplicar controles de aislamiento del ataque con el fin de contener y mitigar el incidente detectado, así entonces se expondrán los avances que actualmente se tienen para las tres contribuciones.

Para la fase de endurecimiento y/o aseguramiento de la red se disponen de múltiples guías que permiten aplicar buenas prácticas de seguridad, generalmente cada fabricante líder del mercado [9] publica en sus sitios oficiales estas guías, por ejemplo, cisco dispone de la “Cisco Guide to Harden Cisco IOS Devices” [10] donde en una metodología por etapas indica al administrador como aplicar los criterios de seguridad, allí se fortalecen variables como: Seguridad en las operaciones, plano de gestión (deshabilitar servicios no usados, gestión de contraseñas, configuraciones de tiempos de espera, alertamiento, monitoreo, SNMP, gestión de accesos, entre otros), plano de control (proxy arp, tazas límites de consumo en el hardware, Protección de protocolos de enrutamiento, entre otros) y plano de datos (controles anti spoofing, inspección arp, seguridad en puertos, protección de vlan, entre otros) [10], así mismo otros fabricantes líderes del mercado como Aruba networks pública su guía ARUBAOS HARDENING GUIDE [11] donde en una metodología similar a la de Cisco especifica criterios de seguridad importantes para aplicarlos a los dispositivos de red. Por otro lado, se tienen buenas prácticas de implementación a nivel de seguridad para todo tipo de infraestructura, un ejemplo de ello es el STIG (Security Technical Implementation Guide) [12] y la NIST 800-53 (National Institute of Standards and Technology) [13] que en ocasiones son usados como criterios de auditoria para validar el nivel de adherencia de los diferentes procesos de implementación de infraestructura en las empresas a los criterios de aseguramiento de la misma, esto debido a que dichas guías

son generales y transversales a todo tipo de marca y/o fabricante. Sin embargo, en la investigación no se logra evidenciar estudios o herramientas que permitan aplicar de manera automática y adaptativa estos criterios de aseguramiento, dado a que muchos de los controles propuestos podrían causar indisponibilidad del servicio si este no es bien manejado y contextualizado.

En la fase de detección es donde más trabajos de investigación y soluciones en el mercado se tienen, por ende, la contribución en esta fase no es mayor, sin embargo, se logra aportar algunos esquemas específicos y particulares para la detección de ataques tipo spoofing, dentro de las técnicas existentes se tienen modos de prevención los cuales analizan directamente el tráfico de red identificando el comportamiento y firmas de seguridad que describen cada una de estas amenazas, Estas técnicas de detección se logran mediante el uso de herramientas IDS (intrusión detection system) [14], que generalmente utilizan dos técnicas de detección de amenazas una por detección de anomalías encontradas a raíz de modificaciones en el comportamiento normal del tráfico de red TCP/IP, este comportamiento “normal” se define ya sea por perfiles configurados o un por medio de un entendimiento previo de la red, un segundo método basado en las firmas que identifican el ataque y el análisis de patrones como; hora, contraseñas, periodicidad, etc [15]. Al final del desarrollo de la tesis se entrega una arquitectura desde donde en un sistema central se analiza las identidades con las cuales las estaciones de usuario están asociadas a nivel de red, a su vez, estas identidades son recolectadas por agentes en cada estación de usuario protegido y posteriormente enviadas al sistema central para su validación.

Finalmente, en la fase de contención y erradicación automática se tienen recursos empresariales las cuales hacen parte del “know how” de cada fabricante, por ejemplo se disponen de soluciones tipo SIEM (Security information and event management) que a raíz de una correlación de eventos toman acciones sobre la infraestructura compatible, un ejemplo de ello es Splunk quien desarrollo el adaptative response [16], McAfee con Automate incident response [17], DarkTrace con su inteligencia de amenazas y respuesta automática Antigena [18] y Fortinet con su ecosistema Fabric y secure Access que automatiza la respuesta a incidentes frente a la detección de amenazas, sin embargo,

estos recursos al ser comerciales conllevan un alto costo en las organizaciones por las que muchas de ellas no podrían adquirir dichas soluciones. Es por ello que esta tesis de maestría entrega como producto final un esquema que permite al sistema central interactuar directamente con las redes de datos, identificar una amenaza tipo spoofing y aislarla de la red automatizando la respuesta al incidente generado .

# Objetivo general.

Crear una arquitectura que permita fortalecer y automatizar la gestión de incidentes de seguridad internos ante la preparación, detección y actuación de ataques ARP, DNS y DHCP spoofing.

# Objetivos específicos.

1. Implementar un escenario de laboratorio físico o virtual que permita realizar las diferentes pruebas que faciliten recrear, validar y evaluar en todo momento los resultados de las diferentes fases de diseño e implementación.

2. Diseñar una estrategia para el aseguramiento de la red LAN, la arquitectura del agente y los métodos de detección y contención del sistema central.

3. Implementar una arquitectura de atención de incidentes basados en ataques spoofing internos, la cual permita preparar la red, detectar los eventos de seguridad relacionados y tomar acciones automáticas de contención y mitigación.

4. Generar un conjunto de pruebas el cual permita evaluar la arquitectura diseñada, tabulando los diferentes resultados esperados contra los objetivos propuestos.

# 1. Marco Teórico y Estado del Arte.

En la primera fase del desarrollo de este capítulo se dará un contexto del modelo actual de comunicaciones IPv4, sus vulnerabilidades y métodos de aprovechamiento que facultan un incidente de seguridad, además, se ilustrara diferentes técnicas que detectan este tipo de amenazas y metodologías que estructuran bajo buenas prácticas la atención de estos incidentes de seguridad.

## 1.1 Modelos teóricos de comunicación IPv4 y sus vulnerabilidades

Con el fin de facilitar el entendimiento del lector frente a las debilidades y a una posible solución de estas, es necesario presentar algunos conceptos básicos relacionados con los esquemas de comunicación, protocolos y técnicas de ataque que aprovechan las debilidades en dichos modelos. Para ello hay que adentrarse en los comienzos y desarrollos de los protocolos de comunicación donde la única necesidad consistía en poder comunicar los diferentes ordenadores, por ello se plantearon modelos teóricos de comunicaciones que son casi tan antiguos como los propios sistemas de información y donde la prioridad en aquel entonces consistía únicamente en romper las barraras físicas que imposibilitaban dicha comunicación, los temas relacionados con seguridad informática se encontraban en un plano inferior, es más, ni se tenía aun visionado posibles problemas relacionados a agentes mal intencionados, tanto que los modelos de comunicación fueron creados por “caballeros y para caballeros”.

Las bases teóricas planteadas en aquel entonces siguen enmarcando el funcionamiento actual de las redes de datos casi tal cual a como fueron diseñadas, un ejemplo bastante radical es el modelo de interconexión de sistemas abiertos (OSI), creado como un modelo de red y publicado en el año 1984 por (Zimmermann, 1980), en este modelo se marcarían las pautas de cómo debe ser la comunicación entre sistemas.

Los cimientos de este modelo fundamentaron las bases actuales sobre los cuales operan las redes de área local en su gran mayoría e incluso de internet en algunos escenarios, dicho modelo plantea la interrelación de 7 capas donde interactúan de manera natural las capas de aplicación en software, de hardware y de características físicas para establecer una comunicación. Ver figura 1.

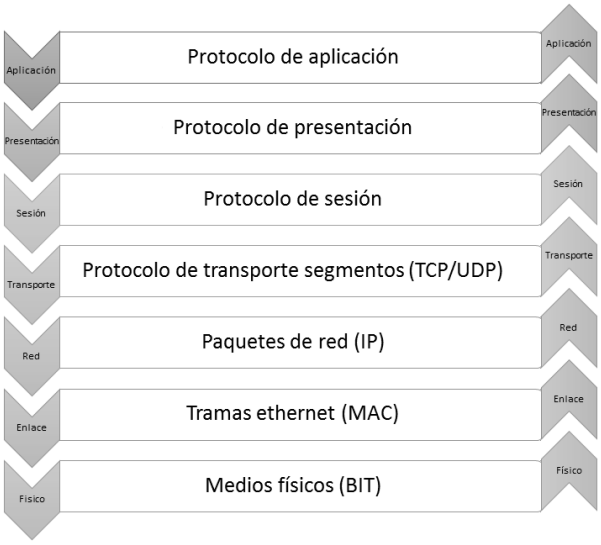


**Figura 1. Modelo de comunicación OSI [16]**

El proceso de comunicación inicia desde las capas superiores (Aplicación, Presentación y sesión) las cuales se encargan de ofrecer un medio entendible e intuitivo al usuario final para que este pueda interactuar de manera fácil y eficiente con los sistemas de información, además, de ofrecer diversas herramientas y protocolos que facilitan muchos los procesos que hoy día sustentan las redes de datos (DNS, DHCP, HTTP, SSH, SMTP, ETC), en la mitad (transporte y red) se tienen las capas que procesan y segmentan las peticiones por medio de protocolos de transporte como TCP/UDP y protocolos de capa de red como IP que direccionan los paquetes hacia su destino, en las capas inferiores (Enlace de datos y físico) se tienen las capas que reciben los encapsulados de las capas superiores y se encargan de poner los datos y encabezados en los medios físicos, esto mediante

direcciones físicas MAC y bits de estados 1s y 0s representados señales de voltaje, haces de luz u ondas electromagnéticas dependiendo del medio en donde se pongan, cuando los datos llegan a su destino el proceso se realiza exactamente igual pero en sentido contrario.

Con el concepto un poco más claro de cómo fueron planteados los modelos teóricos de comunicaciones es posible adentrarse y profundizar en algunos de los ataques spoofing que aprovechan las vulnerabilidades de este modelo y sobre los cuales este proyecto de maestría pretende dar un aporte al automatizar la respuesta a incidentes frente a los mismos.

En los comienzos y desarrollos de los protocolos de comunicación que conformarían los sistemas de información y telecomunicaciones de hoy en día no se tuvieron en mente esquemas de seguridad que permitieran validar la integridad y confiabilidad de las identidades de las estaciones de trabajo que entrarían a intercambiar información, en su lugar se planteó un esquema altamente funcional que cubría las necesidades de aquel entonces, hecho que posteriormente aprovecharían las personas y agentes malintencionados para crear escenarios que permitieran el robo de datos e información importante a partir de la suplantación de identidades usadas por estos protocolos [4].

Hoy en día muchos entornos son vulnerables a estas suplantaciones pues no cuentan con los recursos, conocimientos ni herramientas para mitigar o prevenir las amenazas que aprovechan estas vulnerabilidades, problemática que apoya el desarrollo de la idea central de este documento, para ello se mencionara las suplantaciones y técnicas más ejecutadas a lo largo de la historia.

### 1.1.1 ARP Spoofing o ARP Poisoning.

EL ARP Spoofing o ARP Poisoning, basa su operación en la propia funcionalidad del protocolo ARP ya que en los procesos de comunicación que se establecen entre dos o más sistemas, se hace necesario realizar la unión entre las capas lógicas con las físicas, para ello es indispensable combinar de una forma u otra el valor de identificación en la capa 3 (dirección IP) y en la capa 2 (dirección MAC). De esta forma un paquete puede ser identificado completamente y encaminado correctamente a través de todos los niveles de la comunicación [4]. El protocolo encargado de relacionar la dirección lógica IP y la

dirección física MAC es el ARP y las especificaciones de este datan del año 1982 siendo definidas a través de la RFC 826 [19]. La idea fundamental de este protocolo consiste en la obtención de la dirección física de un sistema conociendo su dirección IP. Para la obtención de dicha información se envía una petición de tipo ARP Request al que le corresponderá una petición de tipo ARP Reply en caso de respuesta afirmativa, cabe resaltar que en el proceso no se realiza ningún tipo de validación en la integridad de las estaciones que solicitan y envían información.

EL ARP Spoofing trata la suplantación de la identidad IP por medio de la falsificación de la tabla ARP de las víctimas, se basa en la construcción de tramas de solicitud y respuesta ARP modificadas con el objetivo de forzar el envío de paquetes de un equipo victima a una estación atacante en lugar de hacerlo a su destino legítimo.

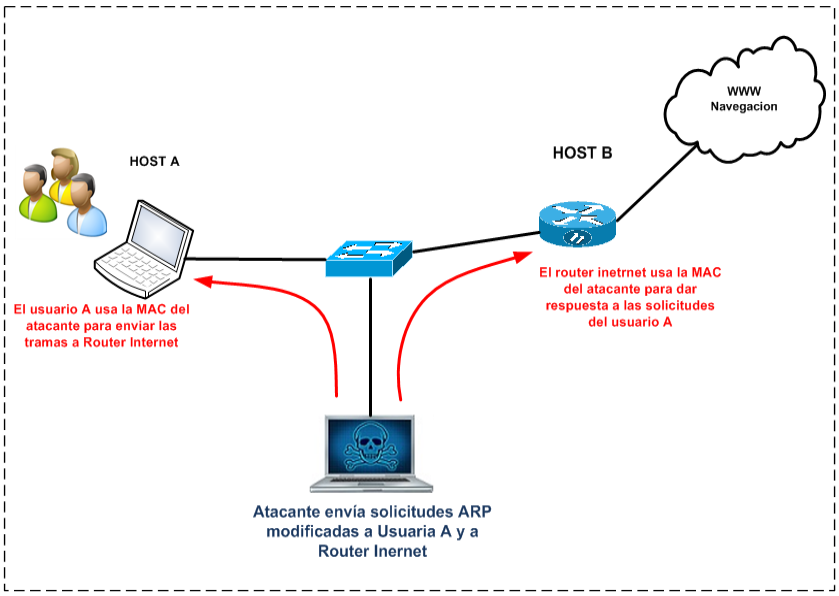


**Figura 2. Ataque ARP Spoofing [18]**

En figura 2 se puede identificar que el tráfico que es originado por el Host A hacia internet es interceptado por el atacante informático generando un alto impacto pues información sensible con algoritmos débiles de cifrado, o incluso, sin ningún tipo cifrado pueden ser interpretados y/o modificados por el atacante, evidenciando este riesgo, se genera la siguiente pregunta:

**¿Qué tan complejo es ejecutar un ataque ARP Spoofing?**

La respuesta es que no son nada complejos de ejecutar pues existen miles de fuentes y referencias que guían en un paso a paso la fabricación de estos ataques, un claro ejemplo es el ejercicio realizado por el instituto nacional de ciberseguridad de España en el taller práctico ARP Spoofing [20] donde se expuso aplicaciones desarrolladas de carácter libre que pueden lanzar el ataque inicialmente obteniendo las direcciones MAC de los hosts victimas (host A que origina el tráfico y host B al que se dirige el tráfico) a través del protocolo ARP que asocia la dirección lógica IP con la física MAC, después se procede con inyectar desde el atacante el paquete malicioso que envenena la tabla ARP de la víctima (Host A) con la MAC errónea del dispositivo al que se dirigirá el trafico (Host B), de este modo el tráfico originado desde el host A seria redirigido hacia el atacante interceptando así los datos.

## 1.1.2 DNS Spoofing.

Otra técnica bastante dañina es El DNS Spoofing que hace referencia al falseamiento de una dirección IP ante una consulta de resolución de nombre, es decir, resolver con una dirección IP falsa a un registro de nombre DNS o viceversa [21]. Esto se puede conseguir de diferentes formas, desde la modificación de las entradas del servidor encargado de resolver una petición DNS hasta comprometiendo un servidor que infecte la caché de otro (lo que se conoce como DNS Poisoning), o empleando técnicas de hijacking las cuales en algunas ocasiones aprovechan el spoofing que además de interceptar el tráfico lo modifican, es decir, un atacante estaría en la capacidad de interceptar una respuesta DNS

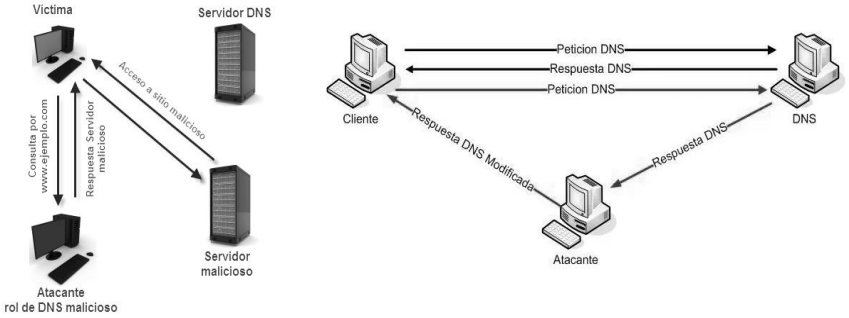


Figura 3. DNS Spoofing [19]

y modificar el registro de modo tal que la estación que está esperando la respuesta reciba una IP modificada al nombre solicitado. Ver figura 3.

En la primera imagen se observa como el servidor DNS fue suplantado permitiendo que la víctima realice todas sus peticiones de consulta DNS al servidor malicioso, en la segunda imagen el atacante intercepta las respuestas DNS con el objetivo de modificarlas a su conveniencia. Esta técnica es usualmente combinada con técnicas de phishing donde se suplanta sitios web de interés con el objetivo de lograr que la víctima ingrese allí sus datos y credenciales.

**¿Qué tan complejo es ejecutar un ataque DNS Spoofing?**

De igual manera que el ARP spoofing existen herramientas libres como Cain y Abel, ettercap, dnsspoof, entre otros que de manera fácil, guiada y efectiva ayudan a construir el ataque y por si fuera poco también existen herramientas de ingeniería social que copian sitios web con el fin de engañar a las víctimas y complementar el ataque DNS Spoofing con técnicas de phishing e ingeniería social [22].

### 1.1.3 DHCP Spoofing.

Es una variante que ha sido ampliamente explotada basado en la implementación de un servidor DHCP falso, ROUGE DHCP, no es un secreto que la mayoría de organizaciones utilizan el protocolo DHCP para asignar dinámicamente los parámetros IP que son necesarios para que una estación de trabajo interactúe con los recursos de red, esta es una funcionalidad que ahorra y simplifica el esfuerzo del administrador de red, sin embargo, el tener todas las estaciones preparadas para que tomen direccionamiento dinámico abre la posibilidad de que si existen dos servidores DHCP en la red las estaciones puedan adquirir parámetros IP del segundo servidor DHCP y no del autorizado por la organización y si entonces este segundo servidor DHCP es uno dispuesto por un atacante para fines maliciosos la organización se estaría enfrentando a serios riesgos y amenazas informáticas. Esto es por causa del funcionamiento propio del servicio DHCP, el cual se describe a continuación [4]:

- Al momento en que una estación ingresa a una red este envía un paquete llamado DHCP Discovery el cual actúa en formato broadcast y tiene como objetivo identificar posibles servidores DHCP que estén preparados para asignar parámetros como dirección IP, mascara de red, ruta por defecto, DNS, entre otros.
- Una vez el servidor DHCP recibe la petición Discovery revisa su disponibilidad de asignaciones para dar respuesta con un DHCP Offer el cual contiene todos los parámetros solicitados por la estación host.
- El cliente o estación selecciona los parámetros que le interesan y realiza la solicitud formal a través de un DHCP Request.
- Finalmente, el servidor atiende la solicitud mediante un DHCP Ack y envía los parámetros al cliente [4].

En el proceso se identifica una debilidad y es el hecho de que el cliente busca recursos DHCP mediante un paquete tipo broadcast en el DHCP Discover, esto crea un escenario de carrera permitiendo que el servidor que primero atienda la solicitud con el DHCP Offer asigne los parámetros IP a la estación.

La realización de este ataque tiene dos variantes que de una u otra forma dan complejidad y efectividad al mismo, una manera es simplemente habilitar un servicio DHCP dentro de la red y que este empiece a entregar parámetros IP corriendo el riesgo a que asigne las mismas direcciones IP que ha asignado el servidor autentico causando conflictos de direccionamiento y permitiendo que el administrador de red identifique de manera más fácil el origen del problema. Otra variante y un poco más compleja es lanzar el ataque de manera que sea casi que imperceptible para el administrador y es ofrecer solo determinado tipo de información de configuración al cliente que solicita los parámetros, esta técnica se conoce como DHCP ACK injection [4]. De esta forma el atacante no tendrá que preocuparse de las IP asignadas o reservadas por el servidor legítimo, solo se facilitarán los parámetros necesarios para interceptar por ejemplo los paquetes dirigidos a la puerta de enlace o plantear la base para un ataque de DNS Spoofing [4].

Los ataques anteriormente mencionados son los más comunes en la suplantación de identidades al interior de una organización y a pesar que salieran a la luz hace muchos años, hoy día son igual de efectivos en muchas organizaciones, es tanto, se podría afirmar que son incluso más efectivos y dañinos ya que se cuenta con diversas herramientas automatizadas, además, de tutoriales públicos que guían al atacante paso a paso en la

fabricación del ataque. En la figura 4 se logra observar cómo las técnicas de suplantación spoofing son empleadas desde hace muchos años, además se evidencia que el conocimiento técnico necesario para emplear una de estas técnicas no es necesariamente muy avanzado, de hecho, después de la ingeniería social es la técnica de ataque más sencilla de ejecutar.

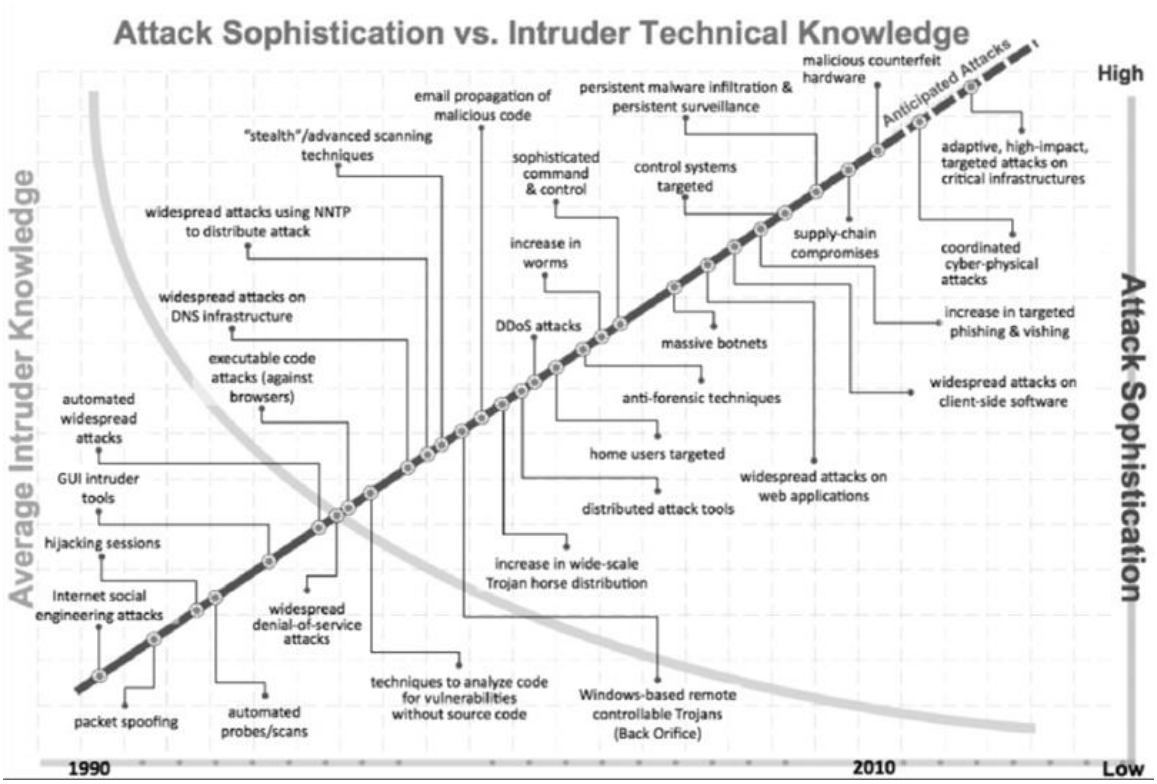


**Figura 4. Sofisticación de un ataque informático vs conocimiento del atacante. [23]**

Contraria y desafortunadamente la identificación y mitigación de estos ataques no es tan sencilla pues requiere de herramientas y métodos que ayuden a dicho propósito, se disponen aplicaciones licenciadas o gratuitas pero que de una u otra manera necesitan de recursos para su mantenimiento, operación y posterior contención y erradicación.

Algunas de estas amenazas pueden ser detectadas por su misma naturaleza, por ejemplo, para el spoofing arp o poisoning arp, se tiene un síntoma que es de relativa facilidad en su detección y es el hecho de encontrar varias IP para una misma MAC.

## 1.2 Técnicas de detección y prevención de ataques spoofing.

**Static MAC Entries**, Esta técnica crea entradas ARP estáticas en los dispositivos evitando que los paquetes de suplantación ARP agreguen dinámicamente entradas en la tabla, sin embargo, dicha medida es compleja de mantener pues exige una administración alta en la gestión de tablas ARP de todos los elementos de la red [24].

**Kernel Based Patches**, se trata parches basados en el kernel del sistema operativo como Anticap o Antidote que han elaborado un mecanismo de protección para el ARP Spoofing a un nivel local. Anticap no permite que una actualización sobre la cache ARP de un host se modifique por una respuesta ARP de una dirección MAC que sea diferente a la que se encuentra registrada en la memoria cache. Antidote valida una Respuesta ARP cuya dirección MAC sea diferente a la previamente registrada en su cache con el fin de comprobar si esta se encontraba aprendida y aún se encuentra activa, si efectivamente la MAC se encontraba registrada y viva Antidote procede a rechazar la solicitud de actualización y registrar la MAC del agresor dentro de una lista de host prohibidos, estas técnicas crean una situación de carrera, es decir si el atacante logra registrar primero su MAC en la victima este quedara como la MAC legitima en la tabla ARP obligando posteriormente al administrador identificar el fallo y proceder con la gestión de solución directamente sobre las aplicaciones [24].

**Passive Detection,** es de las técnicas más usadas con el propósito de identificar ataques spoofing arp pues existen herramientas de fácil implementación y manejo como arpwatch que construyen una base de datos de direcciones MAC con su respectiva IP, de modo que si la aplicación en un futuro detecta un cambio en estas entradas genera una alarma para su respectiva gestión, sin embargo este método pasivo de detección conlleva una debilidad en cuanto al lapso de tiempo entre el aprendizaje de la asignación de la dirección y la detección del ataque, esta debilidad se hace mucho más evidente cuando se manejan grandes volúmenes de entradas en la base de datos permitiendo así que el atacante cuente con el suficiente tiempo de ejecutar la acción maliciosa y escapar [24].

**Secure ARP Protocol (S-ARP),** es una técnica bastante efectiva pues se trata de una "evolución" del protocolo ARP la cual valida una firma de seguridad entre las estaciones involucradas con el fin de garantizar la identidad de ambas partes, sin embargo su implementación exige un cambio total en definición del estándar por lo que es necesario realizar modificaciones en el protocolo de red y las NIC (Network Interface Card) de todos los anfitriones que interactúen entre sí, hecho que dificulta la escalabilidad y gestión de la red [24].

Las técnicas anteriores ilustran diferentes herramientas de detección para ARP spoofing a nivel local de cada anfitrión, sin embargo existen funcionalidades en los dispositivos de capa de enlace que permiten la identificación y mitigación de esta amenaza, un caso puntual la es técnica de arp y dhcp snooping la cual crea una tabla de asociación de direcciones MAC e IP en base al insumo de asignaciones IP del lease DHCP y la conjugación de la funcionalidad arp inspection el cual analiza la solicitud ARP comparándola con la tabla creada por el DHCP-Snooping, en caso de encontrar una irregularidad genera una alerta tipo log [25]. a su vez el DHCP Snooping por si solo proporciona la facultad de mitigar el riesgo de spoofing DHCP. Esta herramienta permite autorizar las interfaces del switch sobre las cuales viajaran paquetes DHCP-Request evitando así falsos servidores [25]. Aunque es una excelente alternativa para minimizar el riesgo de las amenazas de suplantación interna tipo spoofing solo sería posible proteger los anfitriones que hayan sido ingresados a la red por medio de un direccionamiento dinámico.

Como estos métodos y muchos más se logra detectar la ejecución de ataques spoofing pues conociendo el comportamiento y síntomas de los mismos es posible identificar y hasta bloquear muchos de ellos, sin embargo existen otros modos de prevención los cuales analizan directamente el tráfico de red identificando el comportamiento y firmas de seguridad que describen cada una de estas amenazas, Estas técnicas de detección se logran mediante el uso de herramientas IDS (intrusión detection system), que generalmente utilizan dos técnicas de detección de amenazas una por detección de anomalías encontradas a raíz de modificaciones en el comportamiento normal del tráfico de red TCP/IP, este comportamiento "normal" se define ya sea por perfiles configurados o un por medio de un entendimiento previo de la red, un segundo método basado en las

firmas que identifican el ataque y el análisis de patrones como; hora, contraseñas, periodicidad, etc. [14].

De igual manera existen varios tipos de IDS clasificados según su alcance; uno de ellos es HIDS, (host IDS) el cual protege localmente a un host o servidor permitiendo analizar el tráfico con mayor precisión determinando de esta manera qué procesos y usuarios se involucran en una determinada acción, por otro lado, están los NIDS (Network IDS) los cuales analizan el tráfico de toda una red buscando patrones anormales, es necesario ubicarlos estratégicamente ya que operan como un sniffer [15]. Por tipo de respuesta se pueden encontrar los IDS pasivos, estos son aquellos que solo detectan y notifican el incidente de seguridad, sin embargo, no ejecutan ninguna acción sobre la fuente amenazante y por otro lado están los IDS activos que contrario a los pasivos toman acciones de bloqueo o mitigación al sistema atacante, como cerrar la conexión o enviar algún tipo de respuesta predefinida, sin embargo; esta última clase puede impactan negativamente el rendimiento de la red ya que el IDS debe analizar y procesar todo el tráfico que pasa por él [14].

Snort es uno de los IDS más potentes del momento desarrollado bajo el movimiento Open Source y cuyo pionero fue Richar Stallman quien en las décadas de los 80 dio inicio a este proyecto. Open Source plasma una ideología donde defiende la idea de que todo software debería compartir su código fuente de manera que las comunidades interesadas puedan realizar aportes de manera que el proyecto siempre se encuentre en un estado de crecimiento y mejora continua. [26]

Snort fue desarrollado pensando esta ideología lo que le ha permitido una evolución continúa dejándolo a la fecha como uno de los IDS más potentes y apetecidos en el mercado, Snort fue lanzado públicamente en el año 1998 por Martin Roesch e inicialmente llamado “lighight” intrusión detection tecnology y hasta ahora ha conseguido estar a la cabeza, o en una de las posiciones más altas en tecnologías basadas en detección y prevención de intrusos. [26]

Básicamente Snort actúa como un analizador de paquetes basado en librería Libcap creado como parte de la aplicación tcpdump, esto le permite detectar patrones en la inspección sobre los payload en los paquetes, snort ha evolucionado hasta ser considerado un estándar sobre el cual muchas de las soluciones basan su funcionamiento y operación

contando con más de 4 millones de descarga y 400.000 usuarios registrados convirtiéndose en la herramienta de prevención de intrusos más usada a nivel mundial.

Aunque Snort como IDS ofrece potentes herramientas para la detección de ataques de suplantación de identidades se queda corto como solución definitiva frente al aporte de esta tesis, pues en primera medida es necesario contar con administradores de seguridad los cuales a su vez necesitan experticia y conocimiento para llevar a punto la solución, además de realizar un monitoreo y control permanente de los eventos sospechosos encontrados pues la fortaleza de snort está en la detección y no la mitigación y contención, sin embargo, Asmaa Boughrara y Soulimane Mammar en la sexta conferencia internacional de ciencias electrónicas y tecnologías de la información (SETIT) en el 2014 [27] exponen una alternativa que desde una implementación en Snort es posible tomar acciones activas en lugar de pasivas de manera que los ataques ARP Spoofing puedan ser mitigados desde Snort, el articulo presenta la adición de un plug-in de salida el cual tras detectar un intento de ARP spoofing envía a las victimas paquetes ARP Replay legítimos de acuerdo a lo configurado en el archivo snort.conf, este archivo contiene la información integra de las entradas ARP lo que constituía un procedimiento un tanto manual pues solo las entradas ARP que estuviesen definidas en el archivo snort.conf podían ser remediadas, aquella entradas maliciosas que no estuviesen definidas lograrían efectuar el ataque, no obstante, Asmaa y Soulimane proponen una alternativa para automatizar la adición de las entradas legitimas ARP en el archivo.conf, desde la información del servidor DHCP ya que sobre el mismo se encuentra la relación MAC-IP de las estaciones a las cuales se halla asignado parámetros IP.

Esta técnica es bastante efectiva y se acerca a brindar una solución para la detección y contención de ataques ARP Spoofing, otorgando a la organización un esquema de detección y mitigación auto gestionado de manera libre, sin embargo, solo da tratamiento a la técnica de ataque ARP Spoofing que, aunque es la más común y efectiva no enmarca la totalidad del alcance del proyecto.

Marcia Cordero, Myriam Viñamagua y Carlos Garzón en el 2015 sobre la revista GEEKS-DECC-Report publicaron el artículo “Detección y mitigación de ataques ARP Spoof empleando entornos virtualizados”. Allí exponen técnicas de detección directamente desde scripts bajo el intérprete /bin/bash en Linux, muy similares a los desarrollados sobre esta tesis, básicamente el script guarda en una variable el valor de la dirección MAC del default

Gateway y posteriormente compara repetitivamente el valor de esta MAC con la MAC entregada por la tabla ARP, y bajo el escenario donde estos dos registros no sean iguales se determina que existe un ataque de ARP spoofing, sin embargo, para la fase de mitigación utilizan técnicas en donde de manera automática graban el registro ARP estático del default Gateway evitando que este pueda ser modificado por el atacante, esta técnica logra disminuir el riesgo expuesto por las debilidades presentes en el protocolo ARP, sin embargo, quita visibilidad de los inisider threat desde el mismo desarrollo pues el script de mitigación al evitar la modificación de la tabla ARP desvirtúa la estrategia de detección en donde basan dicha detección en el cambio de los registros de la tabla ARP, por lo que si existe un atacante interno y este intenta realizar un ARP spoofing no podrá ser detectado por el sistema desarrollado perdiendo una oportunidad interesante en la caza de estos. [28]

Rajneet Kaur Bijral y Alka Gupta del departamento de computación y ciencia de TI de la universidad de Jammu en India para el 2017 publicación en el International Journal of Advanced Research in Computer Science el artículo “Study of Vulnerabilities of ARP Spoofing and its detection using SNORT” en el que inicialmente tocan de manera tangencial parte del problema planteado de la presente tesis, las vulnerabilidades presentes en los modelos de comunicación actuales y como aun después de muchos años son igual o incluso más eficaces, seguidamente ilustran como desde Snort y la configuración de los preprocessors del archivo snort.conf es posible registrar las estaciones que se deseen proteger, indicando sobre una sentencia de comando su dirección IP y MAC, así una vez snort detecte un ARP Reply con información diferente genera una alerta. Esta puede ser una alternativa que permite detectar este tipo de amenazas, sin embargo, es bastante corta para dar una solución al problema originado por las amenazas tipo spoofing, en primer lugar porque generalmente las soluciones IDS se ubican en el Switch que gestiona el enrutamiento con el fin de tener mayor visibilidad de la red, por lo que el tráfico entre vlans de un mismo Switch de acceso no llegaría hasta el IDS perdiendo total visibilidad de la amenaza y en segundo lugar porque esta alternativa solo ofrece una solución para la detección y no para la mitigación. [29]

Ganga Rama Koteswara Rao y la Dr. R.Satya Prasad en el 2018 publican sobre el International Journal of Engineering & Technology el artículo “Shielding The Networks Depending On Linux Servers Against Arp Spoofing” en donde proponen una alternativa de

solución al problema de ARP Spoofing para la protección de servidores Linux desarrollando algoritmos de detección bajo scripts en bash, estos algoritmos básicamente buscan entradas duplicadas sobre las tablas ARP y en el caso de encontrar una inconsistencia almacenan en una base de listas negras la MAC e IP comprometidas y envía una notificación al administrador vía email y syslog, posteriormente el firewall local lee estas listas negras y bloquea el tráfico del atacante. Este trabajo se encuentra muy cerca a dar una solución al problema de ARP Spoofing, sin embargo, su arquitectura está basada únicamente en la protección de los servidores, pues solo desde el firewall local se genera el bloqueo de la IP del atacante, sin embargo, en una red de acceso de usuarios finales el atacante lograría robar datos sensibles de protocolos como SMB, FTP, IRC, POP, IMAP, SMTP, LDAP, NTLM, etc. [30]

Finalmente Pankaj GARG, en San Diego 2019 registra la patente “USER - SIDE DETECTION AND CONTAINMENT OF ARP SPOOFING ATTACK” No US 2019 / 0058731 A1, que a partir de varios modelos define diferentes escenarios en donde es posible detectar y mitigar un ataque de ARP Spoofing, básicamente desarrolla 4 métodos en los cuales revisa toda actualización de un registro ARP contra la cache ARP registrada previamente, si esta actualización exige un cambio de la dirección IP y su asociación MAC, el equipo victima realiza una solicitud expresa a la IP que solicita la actualización con el fin de registrar la respuesta de la solicitud enviada, de esta manera sobre la tabla ARP solo reposaran los datos solicitados y no los datos enviados desde un envenenamiento de ARP. Aunque la patente en efecto logra cubrir y minimizar el riesgo de un ARP Spoofing, no da el aporte que se requiere sobre esta tesis en cuanto a la automatización de una metodología de atención de incidentes para las amenazas spoofing internas, pues no entrega elementos que permitan preparar la red y posterior notificación para la detección de esta amenaza. [31]

En los párrafos anteriores se ha mencionado diferentes herramientas y técnicas para la detección de ataques Spoofing, sin embargo, esta tesis dentro de su aporte entrega una arquitectura funcional para la atención del incidente que se genera a través de estas amenazas, por ello a continuación, se da a conocer una metodología de atención de incidentes la cual a su vez está en la capacidad de minimizar el impacto generado a las organizaciones frente a riesgos de seguridad materializados al presentar metodologías de respuesta frente a las amenazas detectadas.

## 1.3 Metodología de atención de incidentes de seguridad de la información.

Una metodología de atención de incidentes en una organización se debe establecer para garantizar los siguientes propósitos [8]:

- Fortalecer la infraestructura y la seguridad de los sistemas con el fin de prevenir posibles incidentes o ataques informáticos.
- Crear conciencia en las diferentes personas que interactúan con los sistemas en materia de seguridad y planes de respuesta.
- Restablecer rápidamente los sistemas y servicios a su normal operación frente a un incidente de seguridad.
- Prevenir futuros incidentes, minimizando el impacto causado por los riesgos y amenazas existentes.
- Establecer procedimientos eficaces que permitan detectar, contener, erradicar y recuperar los activos de información que se vean comprometidos ante un incidente de seguridad.

Para lograr los propósitos mencionados se han definido metodologías basadas en un ciclo de vida el cual realimenta constantemente la madurez del modelo [8].


Figura 5. Ciclo de vida en la atención de incidentes de seguridad de la información [8]

La figura 5 muestra las fases comprendidas por el ciclo de vida de atención de incidentes de seguridad de la información donde en la primera etapa se determina que tan y como está preparada la organización para atender los incidentes de seguridad, seguidamente se

definen las políticas y elementos necesarios para detectar y analizar la criticidad del evento o incidente materializado con el fin de precisar la urgencia en su atención, posteriormente se ejecutan las acciones de contención para controlar el incidente, mitigación para reducir el impacto y recuperación para restaurar el sistema a su normal operación, finalmente se tiene la fase post-incidente donde se realiza la lección aprendida frente al incidente ocurrido, esto con el fin de establecer controles que minimice la futura materialización del mismo incidente.

### 1.3.1 Preparación.

En la fase de preparación se deben definir todos los elementos necesarios que alisten el punto de partida del procedimiento de atención de incidentes, dentro de estos elementos se debe tener en cuenta variables como [8]:

- Aseguramiento de infraestructura.
- Concientización de las personas involucradas en el procedimiento.
- Profesionales y equipo experto encargado de atender los incidentes.
- Disponibilidad de recursos necesarios, tales como: sistemas de gestión de seguridad de la información, herramientas de monitoreo, análisis de riesgos y demás.
- Análisis de comportamiento normal de los recursos informáticos.

Una de las variables que debe tener especial cuidado es el aseguramiento de la infraestructura pues allí se establecen criterios que permitirán minimizar posibles incidentes de seguridad en el futuro, para ello se disponen de guías y buenas prácticas de seguridad que facilitan y guían al administrador de TI la implementación y puesta en marcha de los diferentes servicios, un claro ejemplo son las buenas prácticas definidas en la NIST 800-53, DOD 8500, STIG y las publicadas por cada fabricante de dispositivos para su correcta administración. [12]

Para el desarrollo de este capítulo se tomará como referencia un consolidado de los controles establecidos por los modelos anteriormente mencionados, esto con el fin de establecer una línea base transversal a la marca del dispositivo y así lograr asegurar la

infraestructura que ofrecerá los servicios de red a los usuarios internos de una organización.

### 1.3.2 Detección y análisis

En esta fase se definen los procedimientos y herramientas disponibles para la detección de incidentes, además, de establecer criterios que permitan clasificar la criticidad de cada incidente con el fin de instaurar tiempos de atención sobre los mismos [8].

Existen técnicas y herramientas convencionales que han venido evolucionando a través del tiempo las cuales permiten detectar y clasificar los eventos de seguridad, allí podemos encontrar artefactos como IDS (detector de intrusos), IPS (prevención de intrusos), Sistemas de control de acceso (NAC), sistemas de antivirus, dispositivos de protección firewall, entre otros. Cada uno de estos sistemas exige una experticia técnica importante por parte del equipo de atención de incidentes pues es necesario entender y analizar los eventos de seguridad que estos arrojen, además de conocer los métodos de afinamiento que permitan detectar nuevas amenazas y minimizar los falsos positivos.

Para la metodología propuesta en el artículo se plantearán algunas alternativas de detección para los ataques tipo Spoofing internos las cuales a partir de los síntomas evidenciados en las estaciones finales se logre identificar un ataque de suplantación.

### 1.3.3 Contención, erradicación y recuperación.

Esta fase permite establecer los procedimientos necesarios para contener el evento de seguridad con el fin que su impacto no se acreciente, además, de documentar las diferentes estrategias de contención para que su respuesta sea mucho más rápida y eficiente en futuros incidentes, seguidamente se definen las tareas de eliminación del evento que permita borrar cualquier rastro dejado por el incidente y finalmente se procede a recuperar el sistema para que este opere con normalidad [8].

### 1.3.4 Actividad post-incidente.

La actividad post-incidente tiene como finalidad documentar la lección aprendida donde se reflexione por qué el incidente ocurrió y que pudo fallar en la fase de preparación que

permitió a su vez la materialización del riesgo, en este mismo orden de ideas es posibles identificar las cusas del incidente las cuales permitan establecer controles que cierren las brechas encontradas, estas nuevas medidas entraran a alimentar la fase de preparación [8].

Así se ilustra la metodología con la que se estructuro la arquitectura de detección y mitigación para ataques de suplantación internos tipo spoofing, sin embargo, es necesario precisar las herramientas que fueron utilizadas para el desarrollo de la misma.

## 1.4 Herramientas usadas para el desarrollo de la tesis:

Para el desarrollo de las diferentes fases de la tesis se usaron las siguientes herramientas:

### 1.4.1 GNS3.

Gns3 es un simulador de redes que basa su arquitectura en la emulación de sistemas operativos mediante la virtualización de imágenes extraídas desde un sistema de archivos [32], por ello fue la herramienta seleccionada para recrear los diferentes escenarios de ataques spoofing.

### 1.4.2 Cloud Interface.

Las interfaces Cloud corresponden a un recurso importante dentro de GNS3, pues estas interfaces permiten asociar una NIC (Network Interface Card) física o virtual del host local a la topología implementada en GNS3, permitiendo la interacción de los diferentes dispositivos emulados dentro de GNS3 con máquinas virtuales de VMware, Virtual Box o incluso con dispositivos y host físicos conectados al host local que corre la topología del GNS3 [33].

### 1.4.3 QEMU.

Quick Emulator es un virtualizador genérico de CPU distribuido actualmente bajo licencia GPL (software libre), ampliamente utilizada por GNS3 y el cual permite virtualizar a nivel

de línea de comandos diferentes arquitecturas x86 de manera rápida y eficiente, dentro de las características más importantes de QEMU están [34]:

- Aumento de velocidad: algunas aplicaciones pueden correr a una velocidad cercana al sistema emulado en tiempo real.
- Implementa el formato de imagen de disco Copy-On-Write: Se puede crear una unidad virtual multi-gigabyte, la imagen del disco solo ocupará el espacio utilizado.
- Implementa la superposición de imágenes: Se puede mantener una foto del sistema emulado, y escribir cambios en un archivo separado.
- Es posible salvar y restaurar el estado de la máquina
- Emulación de tarjetas de red virtuales.
- El Sistema Operativo emulado no necesita ser modificado o parcheado.
- Control remoto de la máquina emulada a través de una interface VNC integrada.

### 1.4.4 GPG.

Gnu Private Guard es una herramienta distribuida por medio de una licencia GPL, la cual permite cifrar y firmar los archivos o las comunicaciones de un host de manera fácil y versátil a través de una línea de comandos e implementando llaves de cifrado simétrico o asimétrico [35].

### 1.4.5 SNMP:

Snmp es un protocolo simple de gestión de red el cual permite interactuar con los dispositivos de infraestructura de manera sencilla mediante la indagación de variables propias del sistema, los dispositivos tecnológicos ejecutan procesos (agentes) que permiten esta indagación de manera remota [36]

Para una arquitectura SNMP son necesarios 3 componentes ver figura 6:

- Estación de administración
- Agentes de dispositivo o nodo administrado
- Protocolo de administración

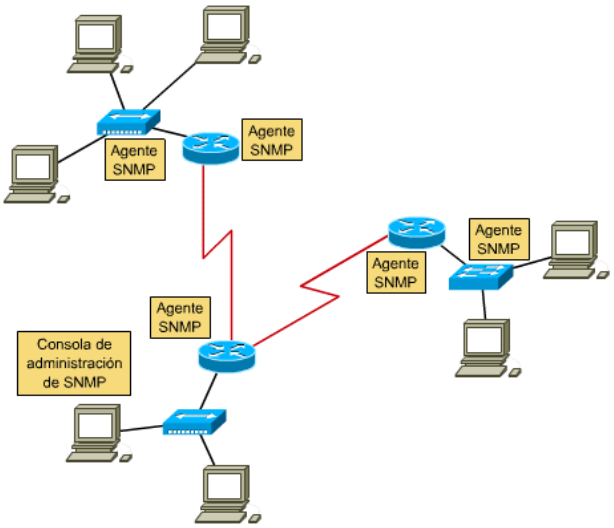


**Figura 6. Arquitectura SNMP. [36]**

Snmp facilita la comunicación entre la estación administradora y el nodo administrado permitiendo que los agentes transmitan datos de variables que se quieran medir o supervisar.

Para la indagación de las variables que se ejecutan en los agentes, SNMP hace uso de bases de datos llamadas MIB (Management information base) las cuales operan como un árbol invertido (es decir con las ramas hacia abajo) donde en sus extremos finales aloja la información deseada, ver figura 7. Estos extremos llevan el nombre de OID (object identification) los cuales a su vez son los usados como punto de referencia para realizar la consulta a través de la estación de administración [36].

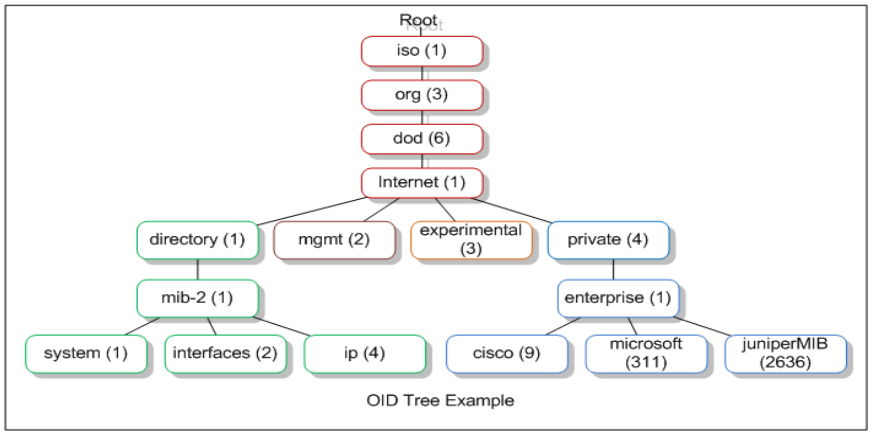


**Figura 7. MIB SNMP. [36]**

**Funcionamiento SNMP.**

Snmp opera básicamente de dos maneras, una mediante pregunta y respuesta (sondeo o Polling) y otra por medio de alertas (trap), cada uno de estos tienen un propósito específico los cuales se mencionan a continuación:

- **Polling:** la estación de administración, lanza una pregunta al agente del nodo administrado con el fin de conocer el estado de una variable que se esté monitoreando, por ejemplo, el estado de la CPU o consumo de red, en base a las respuestas la estación de administración toma accesiones de aletas [36].
- **Trap:** El agente del nodo administrado monitorea constantemente las variables configuradas, en caso de alguna falla o error el nodo a través del agente envía una notificación a la estación de administración para su gestión [36].

**Versiones de SNMP**

Snmp para su funcionamiento utiliza 3 versiones [36]:

- **Snmpv1:** utiliza como mecanismo de autenticación entre la estación de administración y el nodo administrado un stream de caracteres llamado comunidad, este viaja en texto plano sobre la red.
- **SnmpV2c:** funciona como SnmpV1 pero la comunidad viaja cifrada por red.
- **SnmpV3:** la autenticación entre el nodo y la estación de administración es por medio de un usuario y contraseña, la información que viaja por la red es cifrada con algoritmos más fuertes.

## 1.4.6 Nslookup e ipconfig /displaydns:

Nslookup es una herramienta que permite resolver una solicitud de nombre indicando cual sería la dirección IP del servicio solicitado, también es posible utilizar esta herramienta para solicitar una resolución de nombre a través de un servidor DNS externo, por otro lado, el comando ipconfig /displaydns permite listar la cache DNS (Registros dns guardados en memoria). Estas herramientas son bastante útiles para validar la integridad de la resolución Nombre → IP [37].

### 1.4.7 Ipconfig y traceroute:

Ipconfig permite listar los parámetros IP configurados en la tarjeta de red de las estaciones de trabajo, de esta manera es posible identificar y validar si los campos DNS y Default Gateway son los correctos, traceroute permite realizar duna traza de saltos donde se verifique el número de dispositivos que cursa los datos para llegar a su destino, si el número de dispositivos es mayor al definido por la red podría indicar una interceptación de tráfico.

### 1.4.8 GNU Bash:

Bash (Bourne-again shell) es un intérprete de comandos sobre una línea de consola (Shell) para Linux y compatible con sh, la cual ofrece funcionalidades importantes para la programación y uso interactivo con el sistema operativo, generalmente, este interprete en los diferentes ambientes GNU se encuentra ubicado dentro de la ruta /bin/bash. [38]

### 1.4.9 Python:

Python es un lenguaje interpretado y multiplataforma, el cual ofrece programación orientada a objetos, programación imperativa y programación funcional, Python ofrece una serie de ventajas que lo hacen un lenguaje de programación muy atractivo para quienes inician en el mundo de programación o incluso hasta para los más experimentados desarrolladores, dentro de las principales ventajas están [39]:

- Python es un lenguaje muy fácil de comprender: los programas Python son muy compactos, un programa en Python suele ser más corto que otros lenguajes.
- Python es muy legible: La sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta más fácil que si utilizáramos otros lenguajes de programación.
- Python ofrece una interface interactiva, que facilita la realización de pruebas y hallazgo de errores, pues su interface incluye ayudas para ciertas características del lenguaje.
- En entorno de ejecución de Python detecta muchos de los errores en la programación que normalmente se escapan del control de los compiladores.

### 1.4.10 Dynamips y Dinagen:

Dynamips es un virtualizador de hardware para arquitecturas Cisco, que permite emular en arquitecturas x86 diferentes plataformas de la marca Cisco y Dinagen constituye una herramienta de configuración de Dynamips que ayuda con la interacción de los diferentes dispositivos emulados con el hypervisor. [40]

# 2. Metodología.

La imagen de la figura 8 ilustra la metodología que se llevo a cabo para desarrollar los diferentes objetivos de investigación.

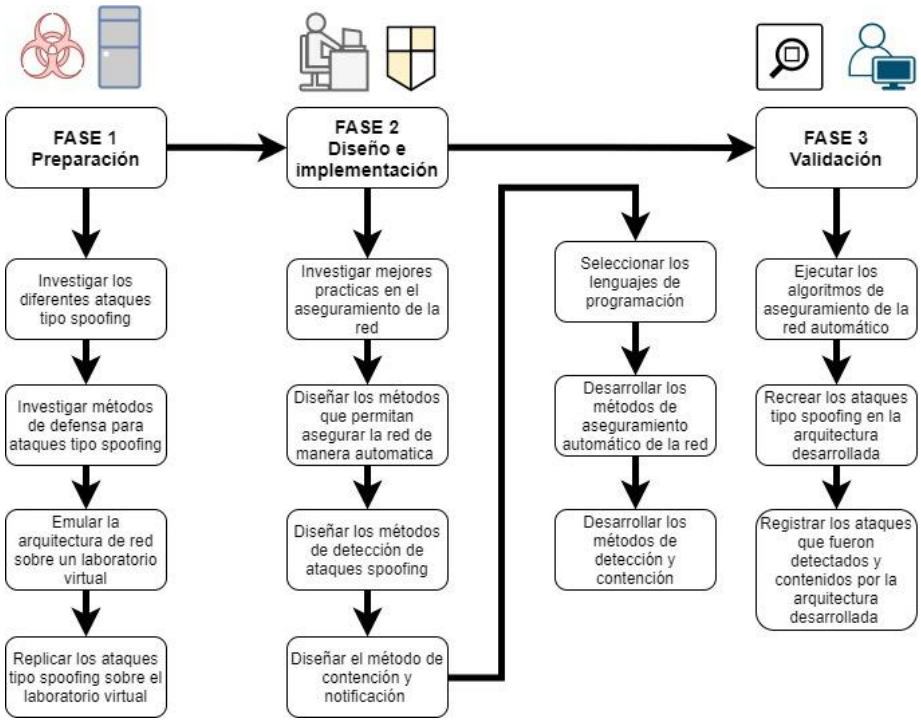


**Figura 8. Resumen metodología**

Para lograr a satisfacción el cumplimiento de los objetivos y entregar las contribuciones propuestas, se estableció una metodología que a partir de 3 fases se entrega la totalidad de los objetivos.

1. **Fase de preparación:** Esta fase tuvo como propósito preparar todos los recursos necesarios para el desarrollo de la tesis, para ello se trabajó el siguiente objetivo.

- (Objetivo1) Implementar un escenario de laboratorio físico o virtual que permita realizar las diferentes pruebas que faciliten recrear, validar y evaluar en todo momento los resultados de las diferentes fases de diseño e implementación

Para lograr este objetivo se realizaron las siguientes actividades

- Se investigó a profundidad los diferentes métodos de suplantación de identidades tipo spoofing que aprovechan las debilidades de los protocolos de red al interior de una organización, además, de las técnicas existentes para la protección de esta amenaza.
- Sobre un escenario físico, real y controlado se replicó los ataques tipo spoofing, esto con el fin de guardar los registros y huellas que dejan cada uno de ellos
- Se investigó posibles soluciones que permiten implementar arquitecturas de red emuladas.
- Se implementó un laboratorio virtual y se replicaron los mismos ataques efectuados en el laboratorio físico identificando posibles diferencias.

Esta fase generó los siguientes entregables, que a su vez determinaron el nivel de cumplimiento del objetivo desarrollado.

a) Un apartado dedicado en el cuerpo de la tesis donde se detalló la fabricación de los ataques spoofing junto con las técnicas de protección implementadas, demostrando por qué las soluciones actuales no dan solución completa al planteamiento del problema.
b) Apartado dentro del capítulo correspondiente a esta fase de la metodología con los hallazgos recopilados de las diferentes actividades donde se concluyó el mejor escenario del laboratorio sobre el cual se desarrollaron las diferentes prácticas del proyecto.
c) Conclusión de las huellas que dejan cada uno de los ataques de manera que se tengan las entradas necesarias que habiliten la fase de diseño e implementación

2. **Fase de diseño e implementación:** Esta fase tuvo como propósito entregar las actividades que permitieron el diseño y la implementación de la arquitectura propuesta, para ello se trabajaron los siguientes objetivos:

- (Objetivo 2) Diseñar la estrategia para el aseguramiento de la red LAN, la arquitectura del agente y los métodos de detección y contención del sistema central.
- (Objetivo 3) Implementar la arquitectura de atención de incidentes basados en ataques spoofing internos, la cual permita preparar la red, detectar los eventos de seguridad relacionados y tomar acciones automáticas de contención y mitigación.

Para lograr estos objetivos se realizaron las siguientes actividades:

- Se investigó políticas y mejores prácticas de endurecimiento para la protección de los dispositivos de red en los diferentes fabricantes.
- Se consolido sobre una única política de aseguramiento mejores prácticas que puedan ser aplicadas transversalmente en diferentes fabricantes.
- Se diseñaron los métodos que permitan aplicar las buenas prácticas a los dispositivos de red sin afectar la disponibilidad de la misma.
- Se diseñó el mecanismo de detección de las diferentes amenazas de suplantación comprendidas dentro del alcance.
- Se diseñó la metodología de contención y notificación adecuada que permitió sanear el problema existente con las amenazas de suplantación.
- Se diseñó un mecanismo de indagación de identidades para dispositivos de red pues sobre los mismos no es posible la instalación del agente desarrollado para esta tesis.
- Se seleccionó dentro de los lenguajes de programación el intérprete Python, el cual permitió materializar los diseños creados sobre módulos de software funcionales.
- Se desarrollaron los módulos e interfaces de aseguramiento para los dispositivos de la red.
- Se desarrollo un módulo de detección y contención que permitió identificar oportunamente la amenaza tipo spoofing y aislarla segundos después de la red.

Esta fase dejo los siguientes registros permitiendo evidenciar el logro de los objetivos trabajados.

a) Apartado dedicado dentro del desarrollo de esta fase de la metodología en el cuerpo de la tesis donde se precise y detalle los elementos necesarios que fueron tenidos en cuenta para proteger la red, el diseño de las técnicas de detección, el diseño del agente y los métodos de mitigación automáticos para ARP, DNS y DHCP Spoofing
b) Diagrama de flujo, código fuente y evidencias que permitieron constatar la aplicación de los diseños sobre un artefacto funcional.
c) Codigo fuente de los diferentes artefactos sobre los anexos del 1 a 5.

3. **Fase de validación:** Esta fase tuvo como propósito verificar que las metas trazadas en las fases anteriores se cumplieron bajo indicadores de eficacia en detección y contención, para ello se desarrolló el objetivo número 4:

- (Objetivo 4) Generar un conjunto de pruebas el cual permita evaluar la arquitectura diseñada, tabulando los diferentes resultados esperados contra los objetivos propuestos.

Las actividades que ayudaron a alcanzar el objetivo trazado fueron:

- Se corrieron en un ambiente real los módulos y algoritmos desarrollados para el aseguramiento automático de los dispositivos de red.
- En un ambiente controlado se recrearon las diferentes amenazas, registrando los hallazgos positivos y no positivos, esta actividad generó un indicador de cumplimiento.
- En un ambiente controlado se recrearon las diferentes amenazas, registrando las que fueron contendías y mitigadas, esta actividad genero un indicador de cumplimiento.

Esta fase dejó el siguientes entregable el cual permite evidenciar el logro del objetivo trabajado.

a) Sobre el capitulo 2.3 se ilustra los diferentes hallazgos que permiten constatar el correcto funcionamiento de los artefactos y procesos diseñados e implementados sobre las dos fases posteriores.

b) El anexo 7 con los logs de notificaicones donde se observa las acciones de detección y contención realizadas por el SCGI

## 2.1 Fase de preparación.

En este capítulo se presenta como se estructuro el laboratorio sobre el cual se ejecutaron las diferentes pruebas que permitieron evidenciar los síntomas que presentan los diferentes sistemas cuando son víctimas de un ataque de suplantación tipo spoofing, este insumo fue la entrada principal para la fase de diseño.

En continúo se explica cómo se estructuro el laboratorio virtual empleando las herramientas GNS3, para finalmente ilustrar como se recrearon los diferentes ataques de suplantación tipo spoofing (ARP, DNS y DHCP) y como se identificaron los síntomas que se presentaron sobre cada una de las víctimas.

### 2.1.1 Implementación y configuración del laboratorio.

Para lograr identificar a satisfacción los diferentes síntomas causados por los ataques de suplantación tipo spoofing se empleó una topología de red básica comprendida por un switch de red, un Gateway (router-firewall) que provee salida a internet y dos estaciones de trabajo identificadas como víctima y atacante. Ver figura 9.

A continuación, se explica de manera general los parámetros con los que fueron configurados el laboratorio de pruebas, además de las configuraciones básicas ejecutadas sobre cada uno de los elementos que compone la red.

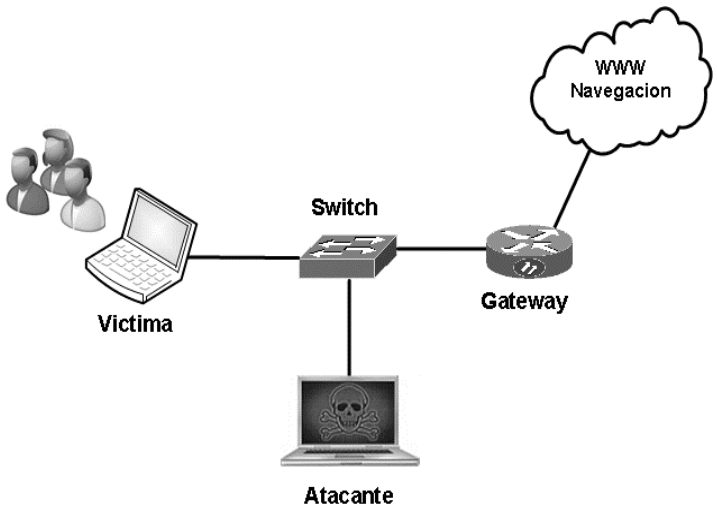


Figura 9. Arquitectura para elaboración del laboratorio.

## 2.1.1.1Configuración del laboratorio.

Para el laboratorio se utilizó los siguientes recursos:

a. Dynamips para la emulación de una imagen de archivos para un router el cual realizara el papel de Gateway para la red.
b. Imagen virtual de un sistema operativo de Windows 7 para simular la figura de la victima
c. Imagen virtual de un sistema operativo Kali 2019 para simular la figura del atacante.
d. Wireshark para visualizar el comportamiento de los datos y el tráfico de red frente a un ataque de suplantación tipo spoofing.
e. Qemu para la emulación de una imagen de archivos Router OS 6.46, correspondiente a Mikrotik.

Finalmente, la topología completamente llevada a GNS3 se visualiza en la figura 10.

## 2.1.1.2 Configuración del router.

El propósito de este dispositivo dentro de la red de pruebas será proporcionar una conexión a internet y gestionar las conexiones capa 3 de los diferentes componentes de la topología,

para ello se usó un dispositivo Router OS de Mikrotik el cual debe ser virtualizado por medio de la herramienta QEMU.

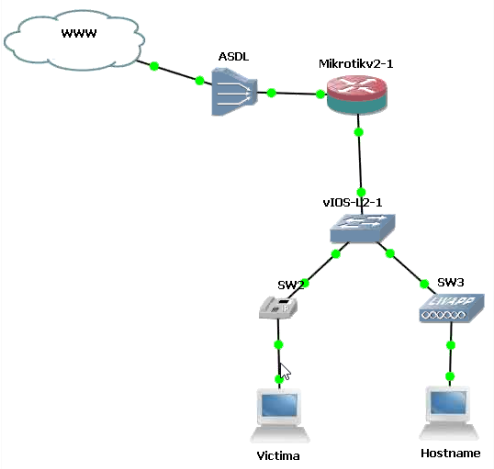


Figura 10. Laboratorio implementado en GNS

**Consideraciones:**

- El direccionamiento asignado para la red LAN fue 10.0.1.0/24 donde la interface del Router fue la ip 10.0.1.1. Ver figura 11
- El direccionamiento WAN fue asignado dinámicamente mediante una interface loopback del sistema operativo, sobre esta interface se comparte el recurso de internet disponible en el equipo físico local, por ello debe estar configurada como DHCP. Ver figura 11
- Se realiza una traducción de direcciones NAT del direccionamiento local al direccionamiento WAN.
- Se configuro un servicio DHCP y DNS para ofrecer los servicios que posteriormente son suplantados.

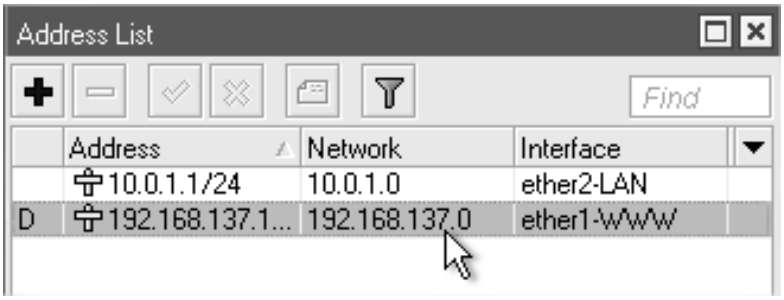


Figura 11. Configuración de direccionamiento IP en Router OS

### 2.1.1.3 Configuración del Switch.

El propósito de este dispositivo es facilitar la conexión de los diferentes elementos de la red y proporcionar visibilidad de las identidades a nivel de capa de enlace, para ello se usó una imagen virtual de un cisco Catalyst.

### 2.1.1.4 Configuración del equipo atacante.

Para este dispositivo se dispone de una máquina virtual en VMware con el sistema operativo Kali Linux.

**Consideraciones:**

- Para incluir la máquina virtual se empleó una interface virtual en GNS3.
- El direccionamiento de la máquina virtual está configurado manualmente. Ver Figura 12.
- Se utiliza el framework de explotación metaesploit.
- Se utiliza las herramientas ettercap y dnschef

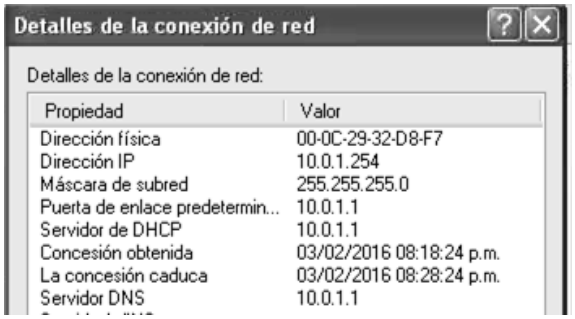


Figura 12. Direccionamiento IP de la interface de la estación victima

### 2.1.1.5 Configuración del equipo víctima.

Para este dispositivo se utilizó un sistema operativo Windows con las siguientes consideraciones:

- El direccionamiento de la estación es asignado por DHCP. Ver figura 13.

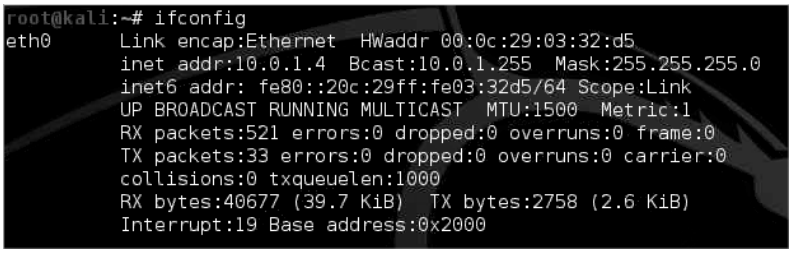


**Figura 13. Configuración de direccionamiento IP en Kali Linux**

Una vez estructurado el laboratorio con los elementos necesarios para realizar la replicación de los ataques de suplantación se procedió a ejecutar los mismos.

## 2.1.2 Replicación de ataques spoofing

Este fragmento del capítulo ilustra como se efectuaron los diferentes ataques de suplantación de identidades tipo spoofing

.

### 2.1.2.1Ataque ARP Spoofing.

Para estructurar el ataque ARP spoofing se inició con la identificación de posibles host victimas desde la red interna, para ello desde la estación del atacante se realizó un escaneo de direcciones IP por medio de la herramienta nmap.

En la figura 14 se observan 3 IP vivas, las cuales corresponden al: Gateway de la red, la víctima y el atacante, este insumo permite a la persona mal intencionada seleccionar sus objetivos

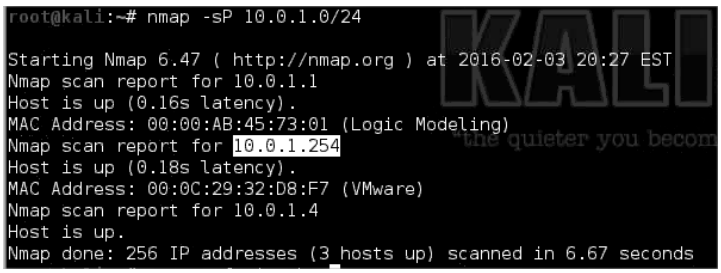


**Figura 14. Scan de la red con nmap desde la estación atacante**

Posteriormente se lanzan los paquetes ARP reply a cada una de las victimas con el fin de envenenar sus tablas ARP y obligar a que el tráfico que se origine de un extremo a otro pase por el host del atacante. Ver figuras 15 y 16.


**Figura 15. Envenenamiento de tablas ARP en el Gateway**


**Figura 16. Envenenamiento de tablas ARP en Victima**

Siendo un ambiente controlado desarrollado con el único objetivo de tomar evidencias se procedió a revisar las tablas ARP de cada uno de los hosts comprometidos con el fin de evidenciar el síntoma de envenenamiento. Ver figuras 17 y 18.

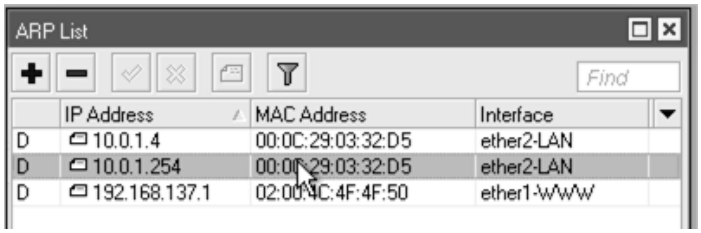


Figura 17. Evidencias tablas envenenadas en el Gateway


Figura 18. Evidencias tablas envenenadas en host victima

Se logra evidenciar claramente que la dirección MAC correspondiente a la IP del Host victima (10.0.1.4) es exactamente igual a la dirección MAC de la IP para el equipo atacante (10.0.1.254), además, si se coteja el resultado con la información de la interface del Kali Linux suministrada en la figura 8 se puede apreciar que esta MAC corresponde al equipo atacante 00:0c:29:03:32:d5, este hallazgo entregó elementos muy importantes para determinar los sitomas y rastros que dejan las victimas de este ataque, los cuales a su vez permitierón diseñar los artefactos de detección, por otro lado en ocasiones es normal ver que la administración de seguridad en las empresas, la gestión de sus aplicaciones internas no sea por medio de protocolos cifrados, por ello para este caso se tomara como punto referencial la gestión del Router OS Mikrotik ejercida por parte del equipo víctima.


Figura 19. Ingreso desde la estación victima a la gestión del Router por HTTP

Una vez la victima ingrese a la gestión del Router el atacante puede observar los datos que viajan entre ambos extremos.

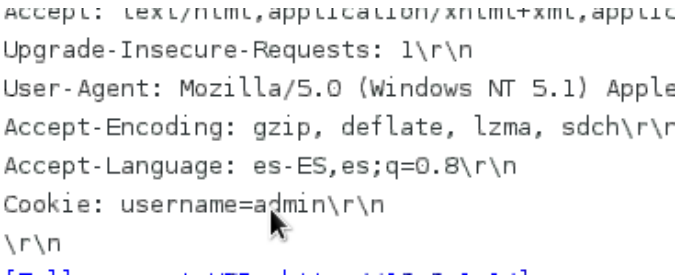


**Figura 20. Visibilidad de los datos de autenticación desde el atacante por medio de wireshark**

En la figura 20 se aprecia como el atacante logra obtener los datos de ingreso a la interface de gestión del Router.

Adicional se realiza la prueba en donde desde la estación víctima se establece una sesión de consola para la gestión remota.


**Figura 21. Sesión de gestión remota Telnet desde la estación victima al Router**

Por medio de Wireshark, el atacante logra tener visibilidad de los datos interceptados.

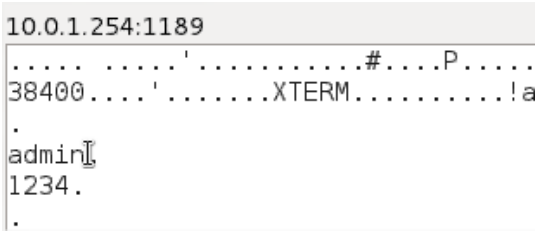


Figura 22. Datos interceptados por el atacante con wireshark

Con los datos recaudados por el atacante es posible establecer una sesión de gestión al router para realizar tareas de post explotación. Ver figura 23.


Figura 23. Ingreso a la terminal del router desde la estación del atacante.

## 2.1.2.2 Ataque DNS Spoofing.

Esta prueba se realizó con base a la intrusión realizada previamente al enrutador, aprovechando que este ofrece el servicio DNS se procede a crear un registro DNS falso con el fin de obligar al usuario a ingresar a sitios inexistentes frente a una solicitud de resolución de nombre, Para el caso de la prueba se tomó como prueba el correo de Google.

En primera instancia desde el equipo atacante se procede a crear el registro DNS falso en el servicio de DNS.


Figura 24. Creación de registro DNS en el Router

Seguidamente de manera controla se realiza una solicitud de petición desde el host victima al registro DNS modificado.

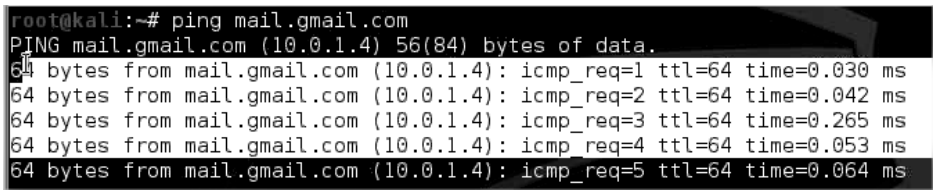


**Figura 25. Resolución de nombre a registro DNS modificado**

Se evidencia que tras intentar resolver el registro DNS modificado por parte de la víctima este apunta al equipo host del atacante, sin embargo, ¿Que beneficio se tiene frente a este hecho? O ¿Que riesgos se tienen?, la respuesta es sencilla siempre y cuando el equipo atacante logre realizar una copia exacta del sitio web que desea suplantar, ya que las solicitudes que se realicen desde el quipo victima serán dirigidas hacia este pudiendo capturar los datos que se ingresen en los diferentes campos, inclusive si estos viajasen normalmente mediante medios cifrados. ver figura 26.

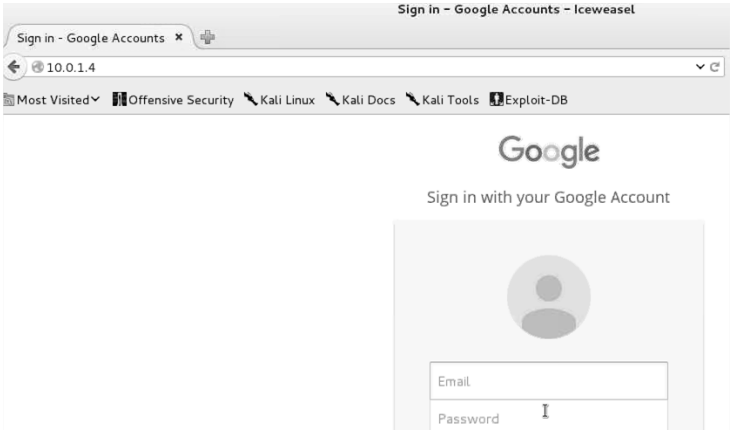


**Figura 26. Evidencia de sitio clonado en equipo del atacante.**

Para ello existen herramientas que permiten clonar sitios web de manera ágil y sencilla, permitiendo que casi cualquier persona pudiese ejecutar este ataque, solo restaría esperar a que la víctima visite el sitio mediante la suplantación de DNS previamente realizada.

Después de las pruebas realizada se evidencia que los síntomas presentados en la victima son mínimos pues el spoofing fue realizado directamente en el servidor siendo casi

imperceptible para los sistemas de información, ya que la resolución de nombres se dio de manera natural.

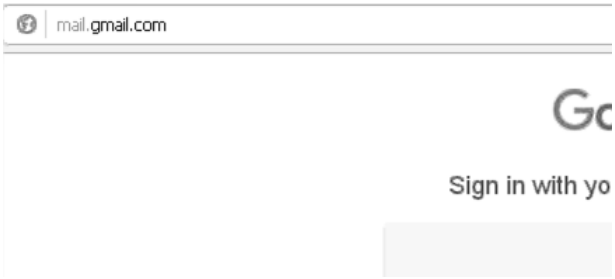


Figura 27. Página suplantada desde el equipo víctima.

### 2.1.2.3 Ataque DHCP Spoofing.

Para este escenario se instauró en el equipo del atacante un servicio DHCP utilizando el módulo auxiliar de metasploit /auxiliary/server/dhcp el cual entrega los parámetros correctos de la red a excepción del servicio DNS, este último corresponde a la dirección IP del equipo atacante. Ver figura 28.

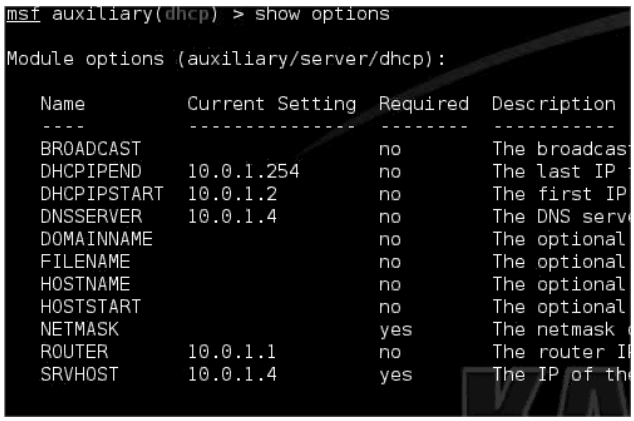


Figura 28. Configuración de servicio DHCP en metasploit

Para lograr el objetivo de la prueba se prepararó el equipo atacante para que este pueda enrutar las solicitudes cuyo destino sea diferente a su IP. Ver figura 29.


Figura 29. Ip forward en equipo atacante

Después se utilizó el servicio dnschef el cual actúa como un proxy DNS, para ello se debe configurar parámetros como:

- Servicio DNS de mayor jerarquía
- Registros DNS a modificar
- IP del registro DNS modificado
- IP donde se aprovisionará el servicio DNS proxy. Ver figura 30.


Figura 30. Configuración del proxy DNS

Finalmente se evidenció que sobre el equipo de la víctima el servicio DHCP haya sido asignado por el atacante permitiendo modificar la IP del servidor DNS y así poder dar continuidad a los ataques de suplantación web anteriormente descritos.

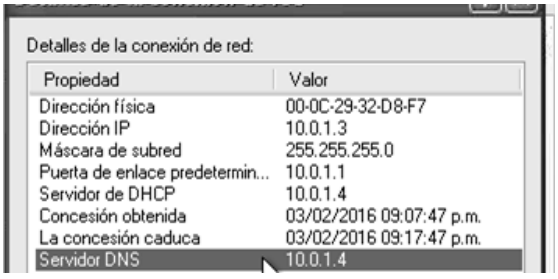


Figura 31. Parámetros IP asignados sobre la victima

En la figura 31 se aprecia los registros modificados por el atacante sobre la víctima, este síntoma entregó un hallazgo simple pero muy eficaz el cual permitió diseñar las estrategias

para detectar este tipo de ataques, pues se evidencia cambios directos sobre los parámetros IP configurados en los adaptadores de red.

De esta manera se da cumplimiento al primer objetivo, implementar un escenario de laboratorio físico o virtual que permita realizar las diferentes pruebas que faciliten recrear, validar y evaluar en todo momento los resultados de las diferentes fases de diseño e implementación, además, de completar el plan de trabajo establecido sobre la metodología de investigación.

## 2.2 Fase de diseño e implementación.

El contenido del presente capitulo ilustra el diseño para la metodología de atención de incidentes que automatiza las fases de preparación, detección y contención buscando atender los eventos de seguridad relacionados con ataques internos tipo spoofing (ARP, DNS, DHCP), para ello se planteó estrategias que preparan la infraestructura de red mediante el aseguramiento en las configuraciones y que es desplegada por un sistema central de gestión de incidentes de ahora en adelante llamado SCGI a la red de acceso, seguidamente se diseñaron las técnicas de detección para identidades maliciosas tanto, directamente desde el SCGI como desde agentes instalados en las maquinas protegidas que detectan los eventos de seguridad y envían la información al SCGI para su gestión, quien toma acciones directas sobre la arquitectura de red para aislar y contener la amenaza detectada.

### 2.2.1 Diseño para la fase de preparación.

Esta fase permite proteger la red frente a posibles intentos de intrusión, no solo a nivel de ataques de suplantación, sino también frente a eventos y brechas relacionadas con el control de accesos y el aprovechamiento de vulnerabilidades de la red, para ello se aplicaron las buenas prácticas definidas en marcos de referencia internacionales como el STIG (security technical implementation Guide) y una secuencia de algoritmos los cuales procesan algunos datos de entrada suministrados por el administrador y los aplica a la infraestructura de comunicaciones ver figura 32.

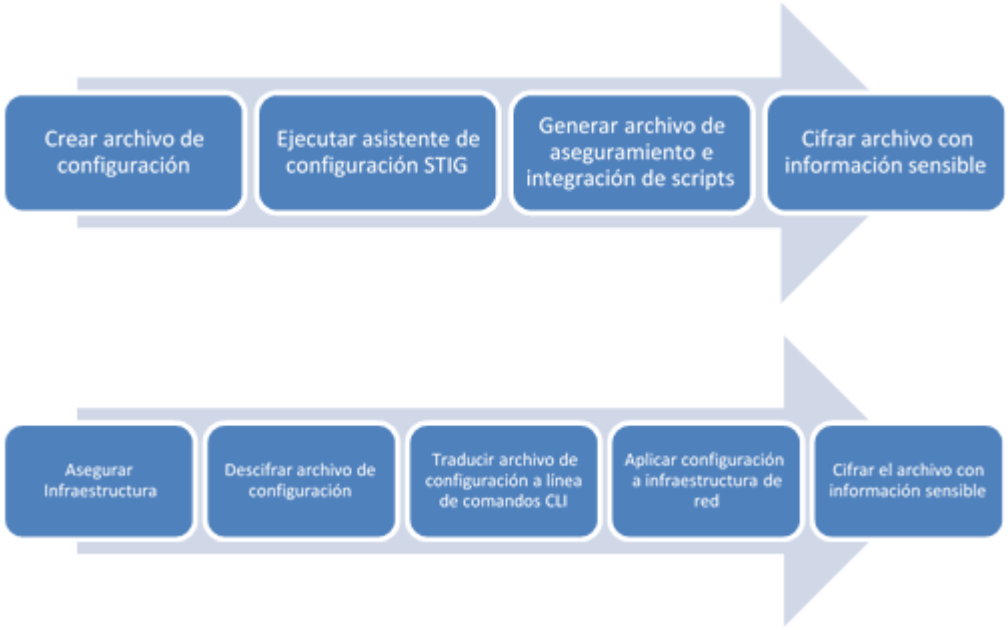


**Figura 32. Artefacto con el que se aplicó el aseguramiento de los dispositivos de red.**

La figura 32 ilustra el modelo diseñado para la fase de preparación construido por paquetes de software, inicialmente se tiene un paquete tipo script que al ser ejecutado lanza un asistente de configuración en donde se solicita al administrador los parámetros de entrada que determinan los datos de configuración, a continuación, el software crea un archivo formateado en texto plano con toda la información ingresada por el administrador de la herramienta, el propósito de este archivo es almacenar las variables que determinan las líneas de comando para el aseguramiento de la infraestructura de red, además, de establecer el protocolo de integración entre los diferentes artefactos del sistema, es decir, sobre este archivo se dispone la información necesaria para interactuar con la red y posterior ejecución de acciones automáticas que aíslen la amenaza spoofing detectada, finalmente sobre esta línea de artefactos se tiene un esquema de cifrado en GPG el cual asegura el acceso al archivo formateado con la información sensible de acceso y aseguramiento de la red.

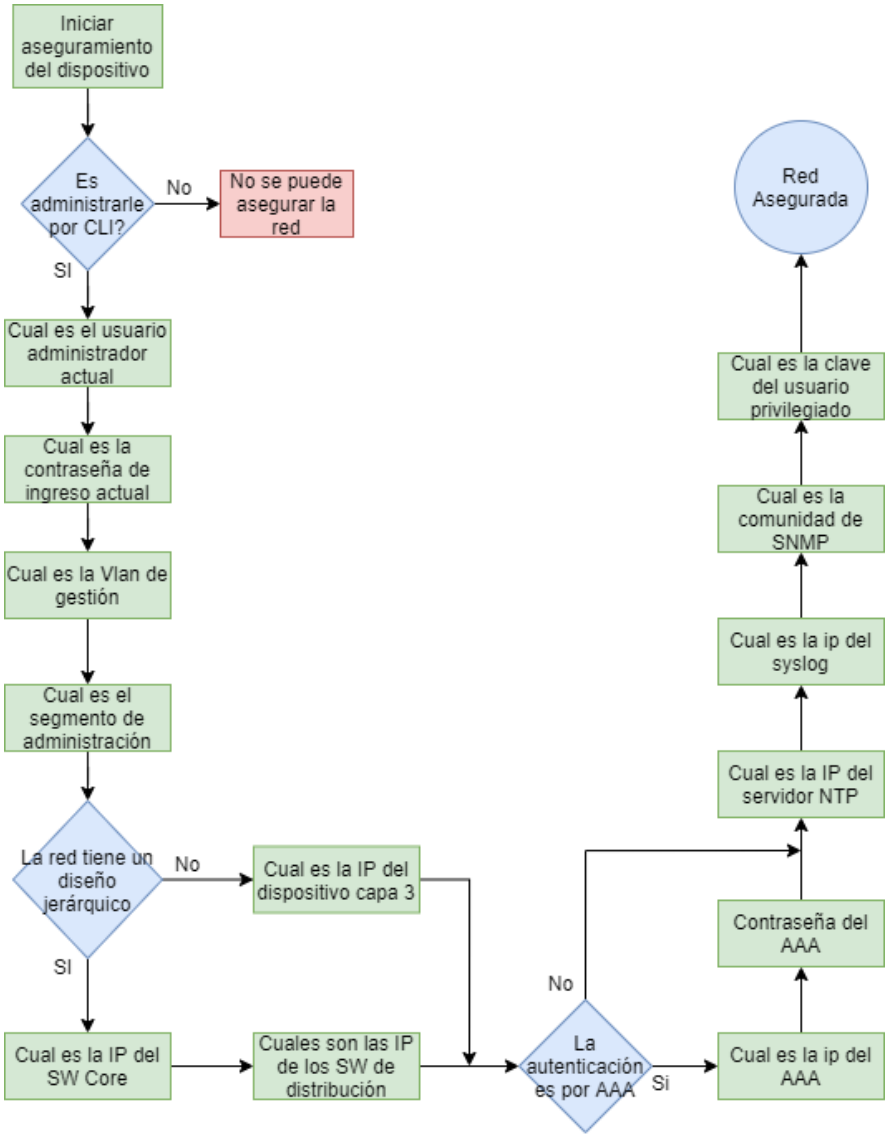


**Figura 33. Diagrama de aseguramiento de red.**

Un segundo script llamado asegurar infraestructura se encarga de descifrar el archivo formateado y traducir la información allí almacenada en líneas de comandos para ser ejecutados posteriormente sobre la infraestructura de red, permitiendo fortificar la implementación y aprovisionamiento de la misma.

Dentro del esquema de aseguramiento de los dispositivos se evidencia una fase crítica la cual da inicio a este proceso, esta fase es de especial cuidado pues requiere de la intervención del administrador de la red, es decir, ante un parámetro mal ingresado se podría generar una mala configuración generando una indisponibilidad de la red, por este motivo y para mitigar el riesgo se debe establecer una guía asistida que permita al

administrador de manera sencilla aplicar las configuraciones de aseguramiento, ver figura 33.

El software para la fase de preparación diseñado permite asegurar la red de datos se encuentra en el sistema central de gestión de incidentes (SCGI), el cual debe tener conectividad con cada uno de los dispositivos de red por medio de protocolos seguros SSH versión 2, esto minimizara el riesgo de interceptación de tráfico en la comunicación del sistema.

Finalmente, para cumplir a satisfacción el diseño de la fase de preapración, fue necesario recoger y consolidar sobre una única línea base las buenas prácticas de implementación que puedan ser aplicadas transversalmente a la infraestructura de red, para ello se definieron los siguientes requisitos desde la guía técnica de implementación de infraestructura STIG:

| Requisito | Debe | Descripción |
|---|---|---|
| V-4582 | El dispositivo de red debe solicitar autenticación para acceder a la línea de consola. | Los dispositivos de red sin contraseña para el acceso a través de la línea de consola posibilitan que cualquier persona con acceso físico al dispositivo pueda realizar cambios de configuración que les permitan interrumpir las operaciones de la red y/o crear puertas de acceso remoto no autorizadas. |
| V-3062 | Los dispositivos de red deben configurarse para garantizar que las contraseñas no puedan ser visualizadas en los archivos de configuración. | Muchos de los ataques informáticos son lanzados desde dentro de la red. Por lo tanto, es imperativo que todas las contraseñas estén encriptadas para que no puedan ser vistas al imprimir la configuración de la consola. |
| V-3210 | El dispositivo de red no debe utilizar las comunidades SNMP predeterminadas o conocidas, públicas y privadas. | Los dispositivos de red generalmente traen configuración por defecto en donde el agente SNMP utiliza la comunidad SNMP public para solo lectura y private para lectura y escritura. |
| V-3043 | El dispositivo de red debe usar diferentes nombres o grupos de comunidad SNMP para varios niveles de acceso de lectura y escritura. | Existen numerosas vulnerabilidades para SNMP; por lo tanto, nombres de comunidad únicos aumenta el riesgo de compromiso dramáticamente. Esto es especialmente cierto con los nombres de comunidad predeterminados de los proveedores que son ampliamente conocidos por los piratas informáticos y otros expertos en redes. |
| V-3143 | Los dispositivos de red no deben tener contraseñas de fabricante predeterminadas. | Los dispositivos de red no protegidos con esquemas de contraseña seguros posibilitan que cualquier persona descifre la contraseña, obteniendo acceso al dispositivo y causando interrupción de la red o denegación de servicio. |
| V-3056 | No deben configurarse contraseñas grupales para el acceso al dispositivo de red. | Las cuentas grupales configuradas en los dispositivos de red no permiten la rendición de cuentas de las personas que usan la cuenta compartida. |

| | | |
|---|---|---|
| V-15434 | Se debe disponer de una cuenta de administración de emergencia para cuando el servidor de autenticación falle | La cuenta de administración de emergencia se configurará como una cuenta local en los dispositivos de red. Se debe usar solo cuando el servidor de autenticación está fuera de línea o no es accesible a través de la red. La cuenta de emergencia debe establecerse en un nivel de autorización apropiado para realizar las funciones administrativas necesarias durante este tiempo. |
| V-5628 | Se debe configurar una VLAN o VLAN de administración dedicada para mantener el tráfico de administración separado de los datos del usuario y el tráfico del plano de control. | Todos los puertos están configurados de manera predeterminada sobre la VLAN 1. |
| V-14671 | Los dispositivos de red deben autenticar todos los mensajes NTP. | NTP es utilizado para garantizar la información precisa del tiempo, NTP podría presentar un riesgo de seguridad si un usuario malintencionado pudiera falsificar la información de NTP. |
| V-4584 | Los dispositivos de red deben registrar todos los eventos excepto los mensajes de depuración y enviarlos a un servidor syslog. | Los registros son una parte crítica de la seguridad. Mantener los registros de auditoría de la actividad del sistema (syslog) puede ayudar a identificar errores de configuración. |
| V-14717 | El dispositivo de red no debe usar SSH Versión 1 para el acceso privilegiado. | SSH Versión 1 es un protocolo que nunca se ha definido en un estándar. Dado que SSH-1 tiene fallas de diseño inherentes que lo hacen vulnerable a, por ejemplo, ataques de hombre en el medio, ahora se considera generalmente obsoleto y debe evitarse. |
| V-5613 | El dispositivo de red debe configurarse para un número máximo de intentos fallidos de inicio de sesión SSH. | Un atacante puede intentar conectarse al dispositivo utilizando SSH, adivinando el método de autenticación y la clave de autenticación. Establecer el reintento de autenticación en 3 o menos fortalece el acceso contra un ataque de fuerza bruta. |
| V-5615 | Los dispositivos de red deben tener habilitado el Keep-Alives para las sesiones TCP. | Las sesiones inactivas TCP pueden ser susceptibles de acceso no autorizado y ataques de secuestro. De manera predeterminada, los switch no prueban continuamente si todavía se puede acceder por TCP de una sesión conectada previamente. |
| V-3078 | Los dispositivos de red deben tener small services TCP y UDP deshabilitados. | Small services, especialmente sus versiones del Protocolo UDP se usan con poca frecuencia. Sin embargo, se han utilizado para lanzar ataques de denegación de servicio. |
| V-3079 | El dispositivo de red debe tener el servicio Finger deshabilitado. | El servicio Finger se utiliza para consultar a un host sobre los usuarios que han iniciado sesión. Si un atacante descubriera quién está usando la red, puede usar prácticas de ingeniería social para tratar de obtener información clasificada de acceso. |
| V-3085 | El dispositivo de red debe tener el servicio HTTP para el acceso administrativo deshabilitado. | HTTP es un protocolo que dentro de su definición no contempla cifrado de datos en la comunicación, por lo que es susceptible a ataques de hombre en el medio (MITM) |
| V-14669 | El administrador debe asegurarse de que los servicios de comando BSD r estén deshabilitados. | Los comandos de Berkeley Software Distribución (BSD) "r" permiten a los usuarios ejecutar comandos en sistemas remotos utilizando una variedad de protocolos. Los comandos "r" de BSD (p. Ej., Rsh, rlogin, rcp, rdump, rrestore y rdist) están diseñados para proporcionar un acceso remoto sin contraseñas a servicios como la ejecución remota de comandos (rsh), inicio de sesión remoto (rlogin) y coipa remota de archivos (rcp y rdist). |

| | | |
|---|---|---|
| V-17822 | La interfaz de gestión no está configurada con ACL de entrada y salida. | La interface de administración debe estar protegida por una lista de host permitidos, minimizando el riesgo que un atacante ingrese al dispositivo desde un host no autorizado |

**Tabla 1. Criterios de aseguramiento para los dispositivos de red**

## 2.2.2 Diseño para la fase de detección.

La fase de detección tuvo como propósito identificar ataques que se realicen dentro de la red interna del tipo ARP Spoofing, DHCP Spoofing y DNS Spoofing, para ello se establece la siguiente arquitectura:

## 2.2.2.1Diseño para la detección por medio de agentes.

Se diseñó un servicio para sistemas operativos Linux el cual integra un artefacto de instalación y tres herramientas de detección para ARP, DNS y DHCP Spoofing.

**Instalación del agente Linux:** para la instalación del agente se diseñó un ejecutable en Linux que en primer lugar solicita al administrador de la red ingresar los parámetros necesarios para realizar la detección de las amenazas tipo spoofing y seguidamente los parámetros para él envió de los hallazgos al SCGI. Ver figura 34


**Figura 34. Diagrama de bloques para la arquitectura de instalación del agente.**

- **Detección de ARP Spoofing desde agente Linux:** el servicio del agente Linux encargado de la detección de ARP Spoofing indaga constantemente la tabla ARP de la estación protegida, buscando encontrar una entrada MAC duplicada para dos o más IP diferentes, esto genera un incidente de seguridad el cual registra un evento bajo un formato especifico donde se relaciona la identidad del Host comprometido, la MAC del equipo atacante, fecha del evento y una marca especial con el fin de que una vez sea enviado el evento al SCGI este reconozca el evento como un incidente que debe ser atendido; finalmente el agente crea un túnel de conexión seguro por medio de SSH y envía el evento al SCGI por medio de un protocolo de copiado SCP (secure copy), para más detalle de la arquitectura ver imagen 35.

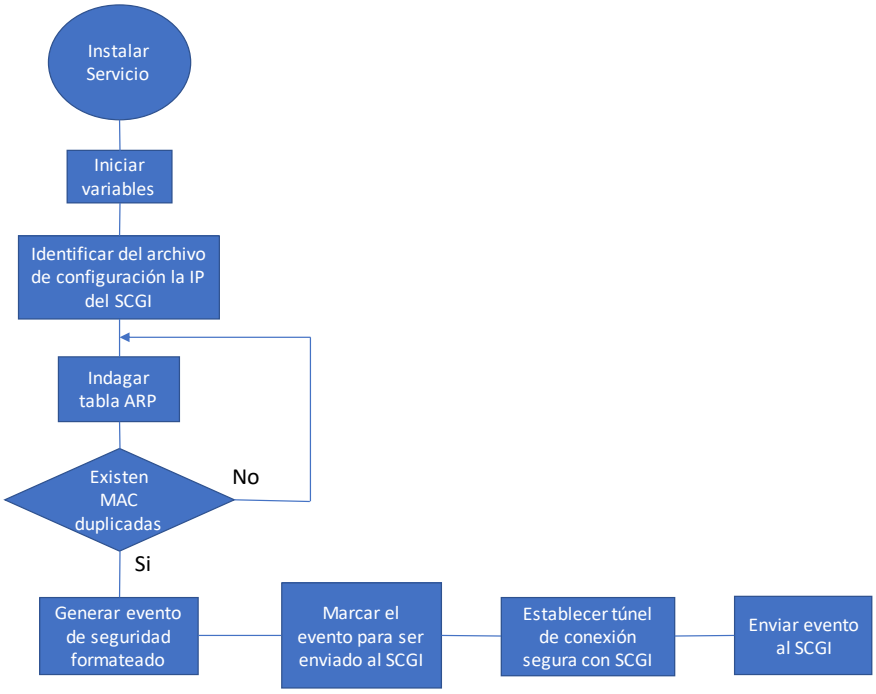


**Figura 35. Diagrama de bloques para la arquitectura de detección de ARP Spoofing**

- **Detección de DHCP Spoofing:** el servicio del agente integra un módulo para la detección de parámetros IP no autorizados, dentro de este se diseñó un método para la detección de puertas de enlace no autorizadas en el segmento de red y la detección de servidores DNS ilegítimos, en este orden de ideas se establece la estrategia para identificar un DHCP falso, entendiendo que el propósito de este protocolo es entregar los parámetros IP correctos a los clientes que lo requieran

para pertenecer a la red, en el momento en que estos parámetros IP sean falseados con el propósito de redirigir el tráfico al host atacante se define que existe un DHCP Spoofing. El agente una vez identifica el hallazgo genera un incidente de seguridad el cual registra un evento bajo un formato especifico donde se relaciona la identidad del Host comprometido, la MAC del equipo atacante, fecha del evento y una marca especial con el fin de que una vez sea enviado al SCGI este reconozca el evento como un incidente que debe ser atendido; finalmente el agente crea un túnel de conexión seguro por medio de SSH y envía el evento al SCGI con un protocolo de copiado SCP (secure copy), para más detalle de la arquitectura ver imagen 36.

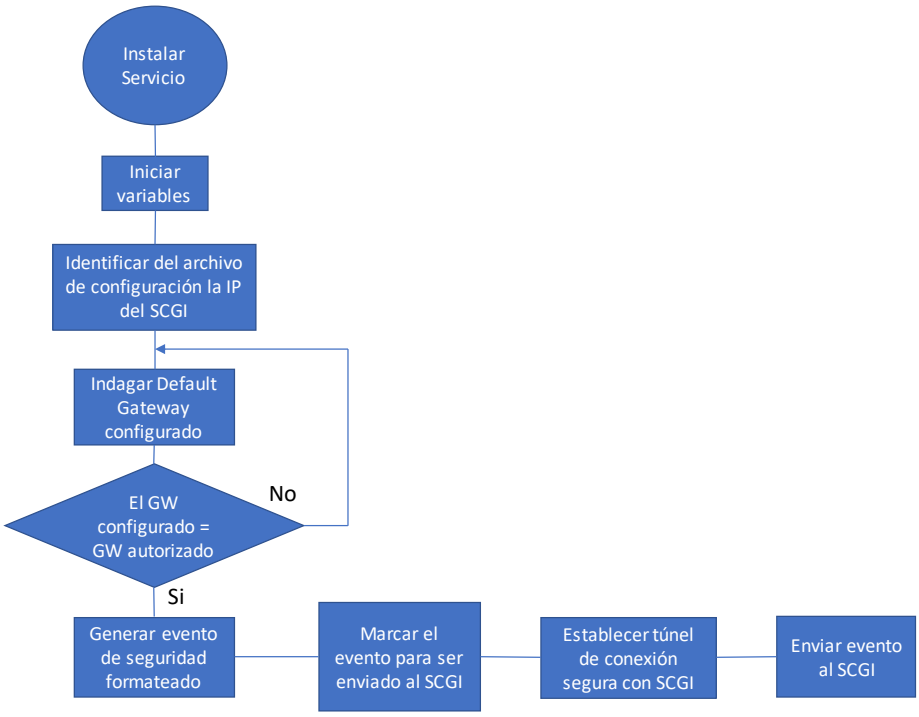


**Figura 36. Diagrama de bloques para la arquitectura de detección de DHCP Spoofing.**

- **Detección de DNS Spoofing:** El módulo de DNS Spoofing ofrecido desde el agente instalado verifica constantemente los servidores DNS configurados sobre la estación protegida y los compara contra los registros DNS del archivo de configuración donde reposan los datos DNS autorizados por el administrador de red, el agente una vez identifica un hallazgo genera un incidente de seguridad el cual registra un evento bajo un formato especifico donde se relaciona la identidad

del Host comprometido, la MAC del equipo atacante, fecha del evento y una marca especial con el fin de que una vez sea enviado al SCGI este reconozca el evento como un incidente que debe ser atendido; finalmente el agente crea un túnel de conexión seguro por medio de SSH y envía el evento al SCGI con un protocolo de copiado SCP (secure copy), para más detalle de la arquitectura ver imagen 37.

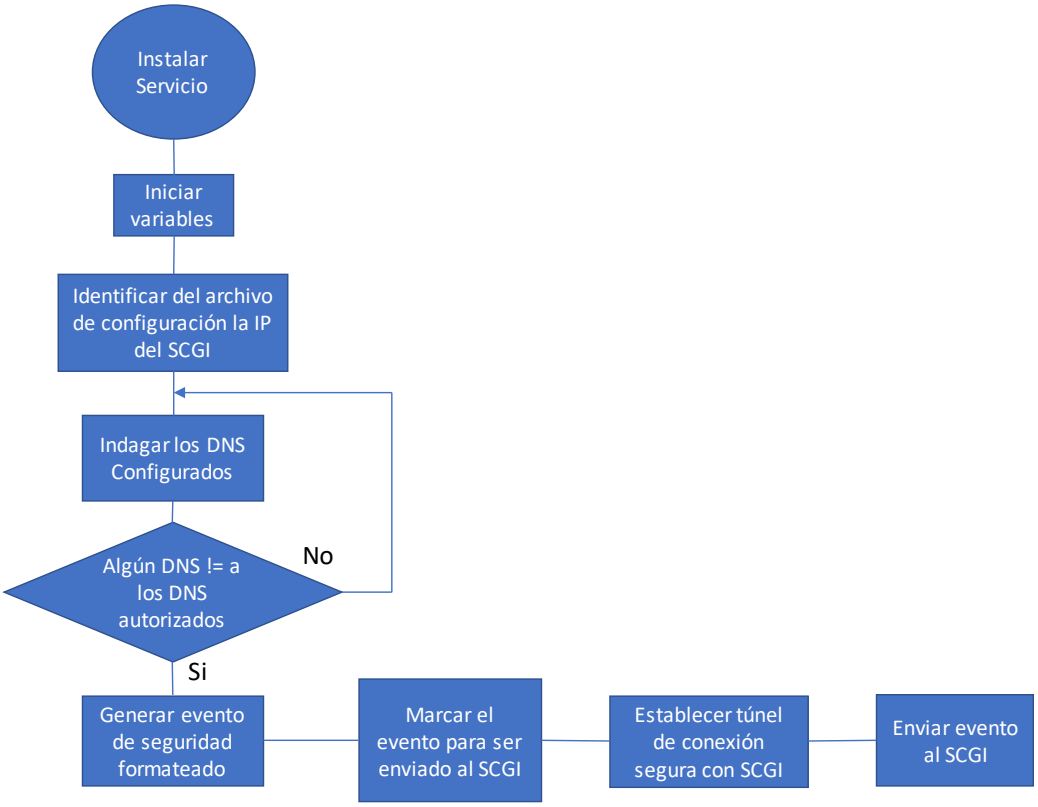


**Figura 37. Diagrama de bloques para la arquitectura de detección de DNS Spoofing.**

En resumen, sobre la figura 38 se muestra la arquitectura de detección de ataques ARP spoofing, DNS Spoofing y DHCP Spoofing.

### 2.2.2.2 Diseño detección de ARP Spoofing desde el SCGI.

Se diseño un servicio que detecta el ARP Spoofing directamente desde el SCGI sin dependencia de los agentes, esto con el fin de proteger los anfitriones de una red de acceso frente a los atacantes que busquen establecer un escenario de hombre en el medio (MITM) por medio de un ARP spoofing, esto genera un aporte importante ya que anfitriones

que pertenezcan a la red bajo un rol de visitante no dispondrán del agente para su protección ya que están por fuera del alcance del administrador de red.

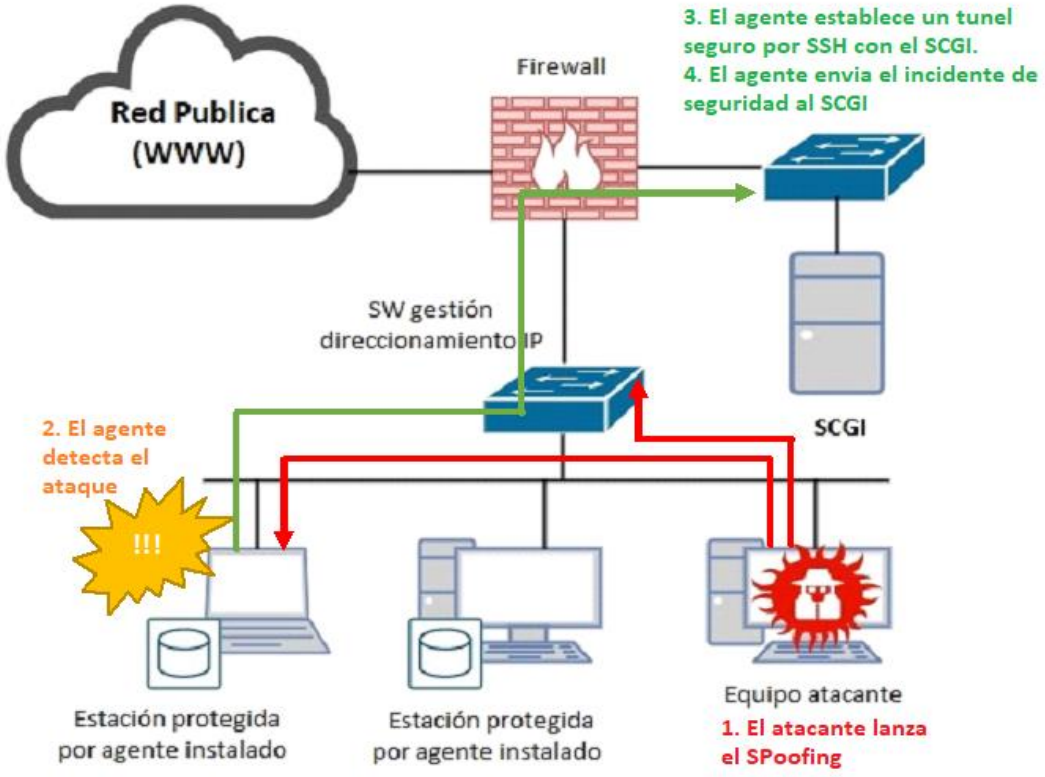


**Figura 38. Arquitectura de detección de ARP, DNS y DHCP Spoofing por medio de agentes.**

Para este método de detección es necesario que el SCGI disponga de privilegios de consulta SNMP al switch o dispositivo que gestione el direccionamiento IP de las diferentes redes o VLAN, ya que por medio de una consulta a la MIB donde se aloja la información de la tabla ARP del dispositivo es posible identificar entradas en donde la dirección MAC se encuentre duplicada para dos o más IPs, en la figura 39 se aprecia la arquitectura de detección de ARP Spoofing directamente desde el SCGI sin necesidad de agentes.

Para este método de detección es necesario crear un servicio en Linux que sincronice los datos del archivo de configuración generado en la fase de preparación, en donde, se cargue la información del fabricante para seleccionar el OID y los datos de configuración del servicio SNMP para así realizar las consultas a la MIB donde se aloja la información de la tabla ARP del dispositivo que gestiona el direccionamiento IP de la red, una vez el servicio sincronice estos datos inicia una indagación constante de la tabla ARP del

dispositivo buscando en sus entradas una MAC con dos o más asociaciones hacia un IP, en la figura 40 se describe el diagrama de bloques para el diseño del servicio.

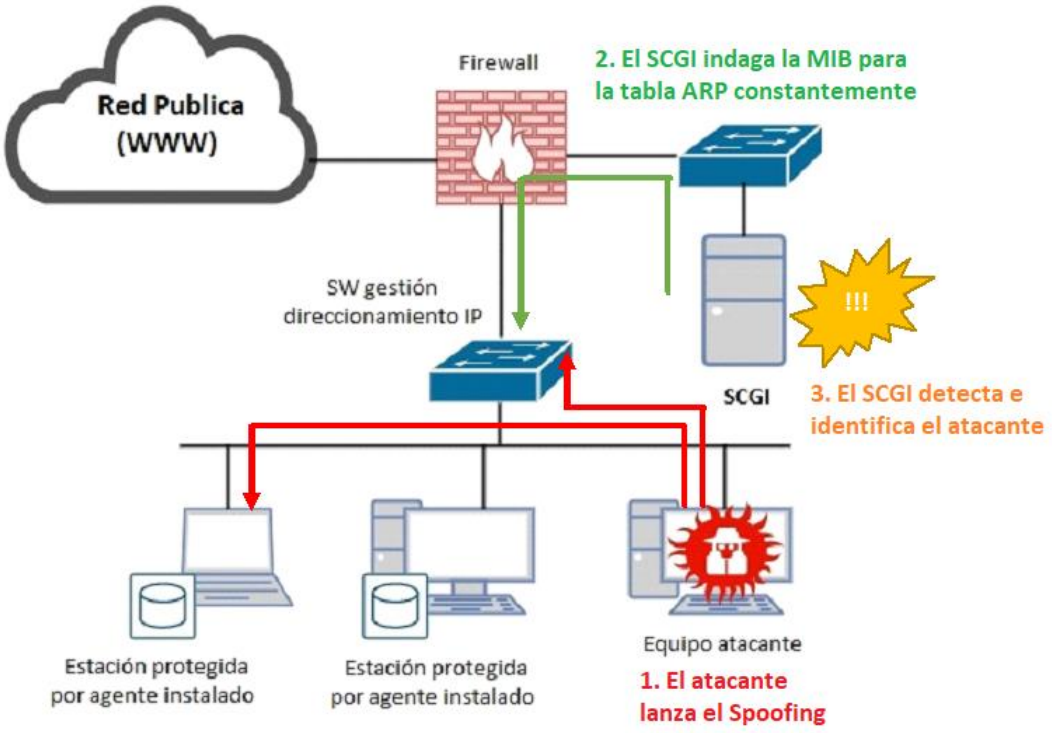


**Figura 39. Arquitectura de detección de ARP Spoofing desde el SCGI.**

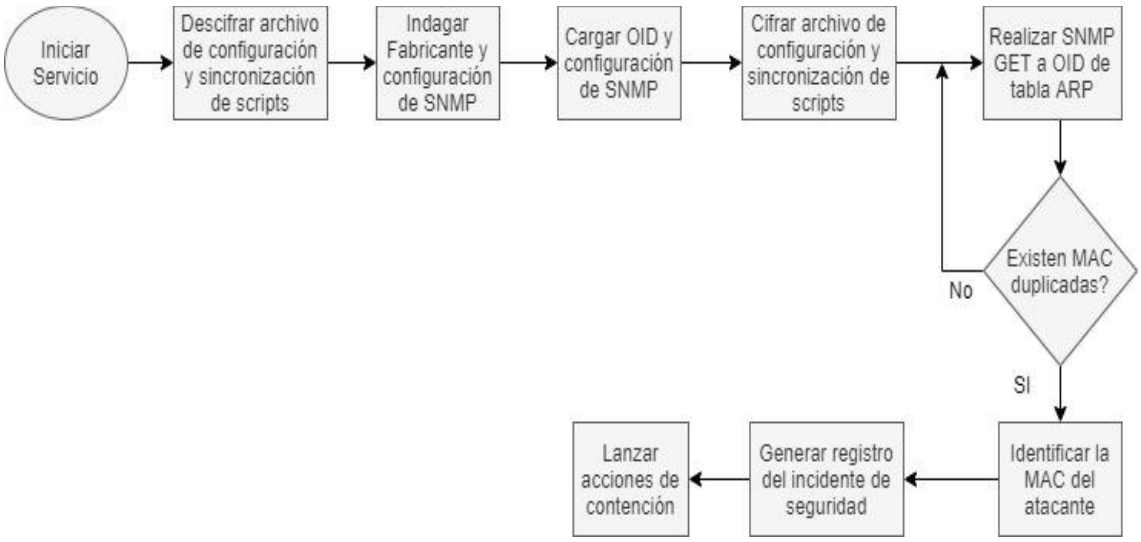


**Figura 40. Diagrama de flujo para la detección y contención de ARP Spoofing desde el SCGI.**

### 2.2.3 Diseño de metodología de contención y mitigación

Esta fase permitirá mitigar automáticamente los incidentes de seguridad detectados en la red interna relacionados con técnicas de ataque spoofing, para ello el SCGI toma acciones directas sobre la red de datos aislando oportunamente la estación mal intencionada, para ello se define un esquema de atención para eventos reportados desde los agentes y otro para los eventos detectados desde el SCGI.

#### 2.2.3.1 Diseño contención y mitigación para eventos detectados por el SCGI.

Para este método de contención se establecio que desde el servicio de detección una vez se encuentre el hallazgo se hace el llamado a la función de contención, en la figura 40 se observa que una vez el servicio encuentra una MAC duplicada para dos o más IP realiza el proceso de registro del incidente de seguridad y posterior invocación de la función de contención.

El proceso de contención se diseñó bajo el lenguaje Python, esto po

r su versatilidad y fácil integración con las diferentes funciones del sistema operativo, en la figura 41 se observa el diagrama de bloques para cumplir dicho fin.

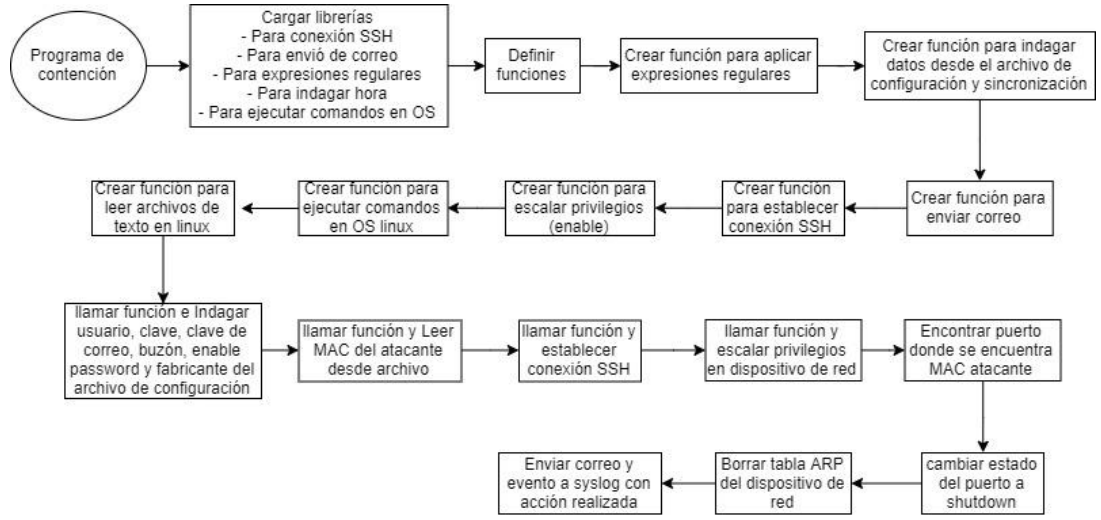


**Figura 41. Diagrama de flujo para la contención de Spoofing**

### 2.2.3.2 Diseño contención y mitigación para eventos reportados desde los agentes

Para este método de detección se diseñó un servicio que constantemente revisa el estado de las alertas reportadas desde los agentes y en caso de encontrar algún hallazgo ejecuta las acciones de contención y notificación correspondientes para el incidente encontrado, en la figura 42 se observa el diagrama de flujo para esta operación.

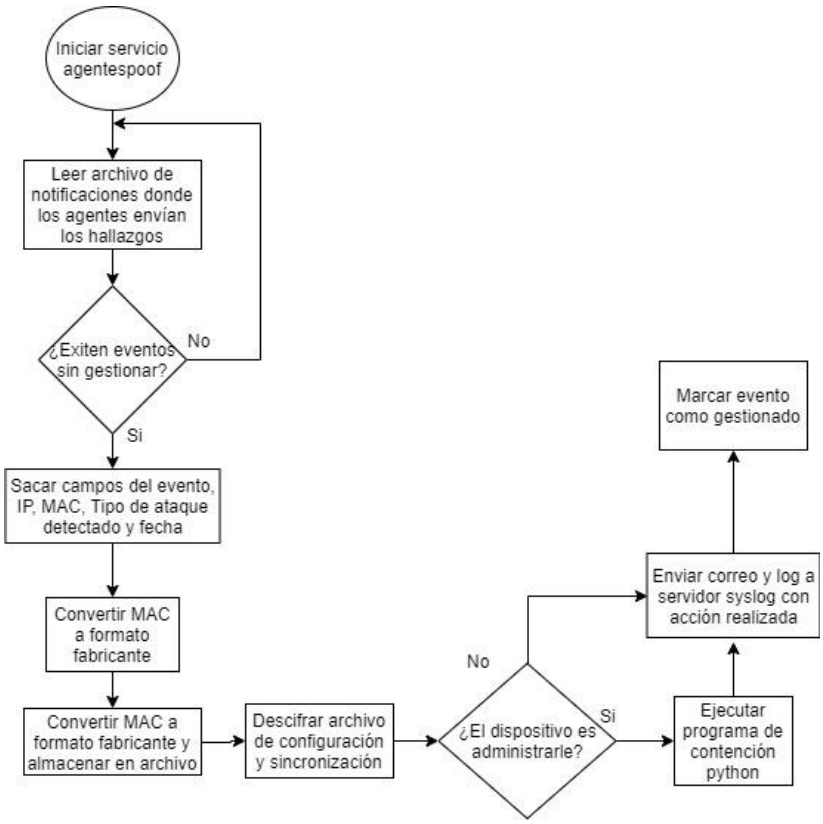


**Figura 42. Diagrama de flujo para la contención de Spoofing reportado desde los agentes.**

El servicio de contención de amenazas reportadas desde los agentes se diseño a manera de script bajo el intérprete /bin/bash, esto con el fin de interactuar directamente con las funciones del sistema operativo Linux donde corren todos los servicios del SCGI, para ello se diseñarón rutinas que indagan constantemente el archivo de eventos donde se almacenan los registros generados por parte de los agentes, estos eventos cuando se envían desde los agentes vienen marcados con un FLAG (bandera) que le indican al SCGI

la llegada de un nuevo evento que requiere ser gestionado y asi una vez el servicio alojado en el SCGI detecta el evento marcado con el FLAG

identifica el tipo de ataque que fue reportado, desde que IP se reportó, la fecha del evento y cuál es la ip y mac del atacante, seguidamente acondiciona los datos para realizar la integración con el fabricante de la solución de networking y así finalmente ejecutar la rutina de contención en Python y posterior notificación del incidente mediante correo electrónico y envió de log a servicio de syslog server.

De esta manera se logró una arquitectura funcional de atención de incidentes de seguridad para eventos tipo spoofing, donde en una primera fase se aseguran las configuraciones para los swtich de la red de acceso, en una segunda fase se establecen los métodos para la detección de ataques arp spoofing, dns spoofing y dhcp spoofing y finalmente sobre una tercera fase se actúa sobre la detección aislando la amenaza de la red dando cumplimiento al objetivo número 2. Diseñar una estrategia para el aseguramiento de la red LAN, la arquitectura del agente y los métodos de detección y contención del sistema central.

### 2.2.4 Desarrollo e implementación de una arquitectura de atención de incidentes basados en ataques spoofing internos.

Este capítulo tiene como propósito presentar los algoritmos, arquitectura, funciones, servicios y scripts desarrollados para lograr el objetivo de la tesis, crear una arquitectura que permita fortalecer y automatizar la gestión de incidentes de seguridad internos ante la preparación, detección y actuación de ataques ARP, DNS y DHCP spoofing. Para ello se explica el detalle de cada módulo y servicio diseñado sobre el capítulo anterior, además, de los lenguajes, funciones y algoritmos utilizados para tal fin, así mismo y con el objetivo de facilitar la comprensión del lector se abordará cada implementación en el orden presentado sobre la fase de diseño, es decir, fase de aseguramiento de la red, fase de detección de ataques spoofing y fase contención.

Es de aclarar que cada uno de los servicios del SCGI se diseñó para un sistema operativo Linux, específicamente se utilizó la distribución de Linux Ubuntu 18.04.2 LTS y el entorno de desarrollo integrado (IDE) Eclipse para el compilador Pydev quien habilita el entorno de

desarrollo para lenguaje Python, esto con el fin de dar un mejor manejo a los componentes de software que se crearon en el transcurso del proyecto. En la figura 43 se observa la arquitectura del desarrollo para los servicios del SCGI.

En la arquitectura de la figura 43 se observa que desde scripts .sh y servicios .service se hacen los diferentes llamados a los módulos que realizan la preparación, detección y contención de ataques tipo spoofing, adicional se logra apreciar que todos los módulos de Python (.py) hacen uso del módulo Funciones_tesis.py, este componente de software tiene una característica especial pues su único propósito es ofrecer las diferentes funciones que son llamadas desde el resto de módulos Python. El resto de componentes se explican de manera general ya que en párrafos siguientes se revisa el detalle y el código de cada uno de estos.

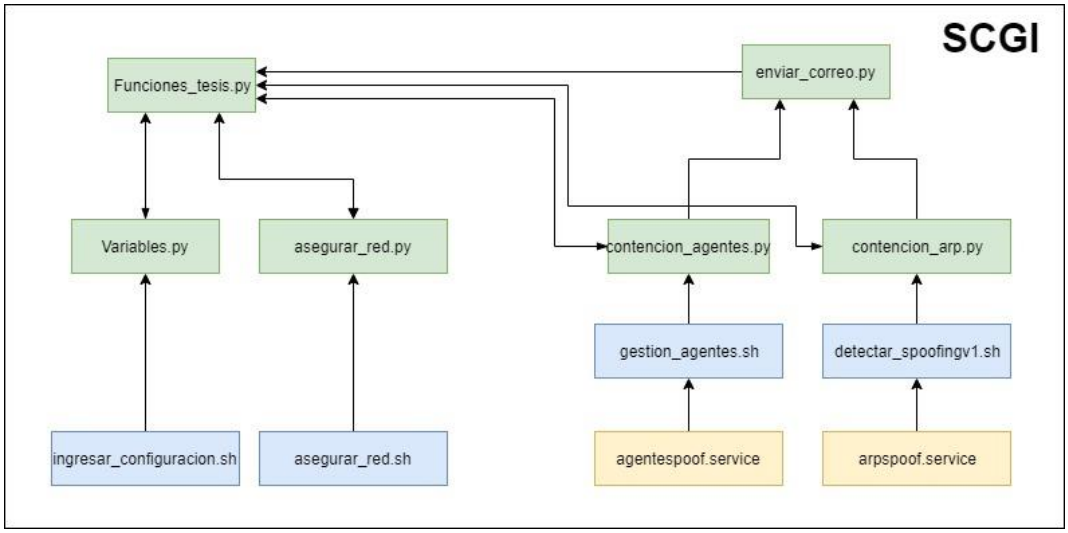


**Figura 43. Arquitectura de software para el servicio de SCGI.**

- **Ingresar_configuracion.sh:** ejecutable para llamar el módulo variables.py
- **Variables.py:** Modulo de Python que por medio de una serie de preguntas al administrador se crea el archivo de configuración con las buenas prácticas del STIG y sincronización con otros scripts y módulos de Python.
- **Asegurar_red.sh:** Ejecutable para llamar el módulo asegurar_red.py
- **Asegurar_red.py:** Modulo de Python que traduce el archivo de configuración en comandos para ser ejecutados posteriormente por el mismo modulo sobre los dispositivos de la red de acceso.

- **Agentespoof.service:** servicio encargado de ejecutar el script gestion_agentes.sh cada vez que inicie el sistema.
- **gestion_agentes.sh:** script encargado de validar constantemente los hallazgos reportados desde los agentes en las estaciones protegidas.
- **Contencion_agentes.py:** Modulo de Python que aísla la amenaza detectada por los agentes sobre la red de acceso.
- **Enviar_correo.py:** Modulo de Python que envía el correo al administrador con las acciones realizadas para la detección y contención de la amenaza.
- **Arpspoof.service:** servicio encargado de ejecutar el script detectar_spoofingv1.sh cada vez que inicie el sistema.
- **Detectar_spoofingv1.sh:** Script encargado de realizar la detección de ARP Spoofing directamente sobre la red de acceso sin dependencia de los agentes.
- **Contencion_arp.py:** Modulo de Python que aísla la amenaza detectada por el SCGI sobre la red de acceso.

**Funciones creadas y ofrecidas desde el módulo funciones_tesis.py**

Desde el módulo de funciones_tesis.py se ofrecen diferentes funciones que son llamadas y usadas por el resto de módulos de Python, de esta manera el código sobre cada uno de los componentes es más liviano y optimizado, en la figura 44 se muestra la arquitectura de este módulo.

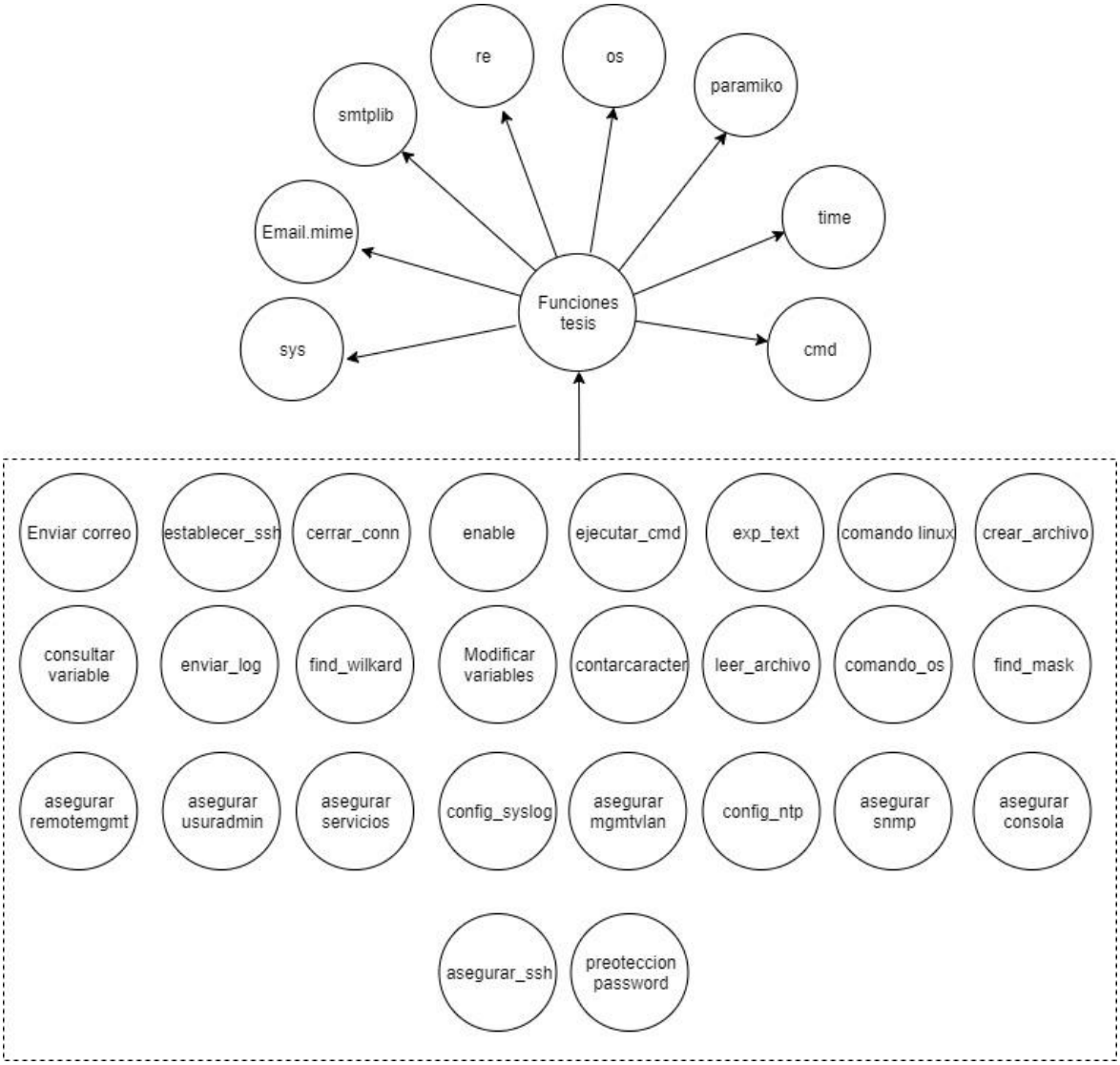


**Figura 44. Modelo de funciones**

Inicialmente se hacen los llamados a otros módulos de Python que son necesarios para la creación de los módulos propios del SCGI, estos módulos son los siguientes:

- **Sys:** Este módulo proporciona acceso a algunas variables utilizadas por Python y a funciones que interactúan fuertemente con el intérprete. [41]
- **Smtplib:** El módulo smtplib define un objeto de sesión para un cliente SMTP que permite enviar un correo a cualquier host de Internet.
- **Re:** Este módulo proporciona operaciones de coincidencia para expresiones regulares.

- **Os:** Este módulo proporciona una forma versátil de utilizar funcionalidades del sistema operativo directamente desde Python. Leer o escribir un archivo, manipular rutas, leer todas las líneas en todos los archivos, crear archivos y directorios temporales, manejo de archivos y directorios de alto nivel, etc.
- **Paramiko:** Paramiko es una implementación del protocolo SSHv2 en Python, que proporciona funcionalidad tanto de cliente como de servidor. Si bien aprovecha una extensión Python C para la criptografía de bajo nivel, Paramiko es una interfaz pura de Python en torno a los conceptos de redes SSH. [42]
- **Time:** Este módulo proporciona varias funciones relacionadas con el tiempo.
- **Cmd:** cmd proporciona un marco simple para escribir intérpretes de comandos orientados a líneas de comandos.

Adicional en la figura 44 se aprecia las diferentes funciones que son propias del SCGI y desarrolladas para ser llamadas por el resto de módulos.

Las funciones de Python son bloques de código que realizan una operación determinada y devuelven un valor o alguna tarea específica, para su creación se debe comenzar con la palabra reservada def, seguidamente escribir el nombre de la función y luego colocar entre paréntesis los parámetros de entrada de la función que se desea crear. [43]

Funciones_tesis.py pone a disposición las siguientes funciones:

- **enviar_correo(msgFrom,msgTo,mail_password,message,msgSubject):** Esta función permite enviar un correo vía smtp, para su llamado se debe indicar en las variables: msgFrom el destinatario del correo, mail_password la contraseña del buzón de correo, en message el mensaje que se desea enviar y en msgSubject el asunto del creo.
- **establecer_ssh(ip,usuario,password):** Esta función permite establecer una conexión ssh invocando una consola de interacción para ejecutar comandos desde el cliente creado, esta función retorna sobre una variable objeto llamada connssh la consola virtual de interacción CLI (Interface de línea de comandos), para su correcto uso dentro de las variables: ip se debe ingresar la IP del servidor ssh, en usuario y contraseña las credenciales de acceso del dispositivo por ssh.
- **enable(enable_password):** Para utilizar esta función se debe establecer previamente una sesión ssh y permite habilitar el modo usuario privilegiado para

dispositivos cisco, su uso solo requiere que en la variable enable_password se indique la contraseña de activación del usuario privilegiado.

- **ejecutar_cmd(comando):** Para utilizar esta función se debe establecer previamente una sesión ssh y permite lanzar comandos sobre una consola de interacción CLI (Interface de línea de comandos), esta función retorna sobre la variable resp la respuesta del servidor ssh frente al comando ingresado, para su uso dentro de la variable comando se debe indicar la línea de comando que se desee aplicar.
- **cerrar_conn():** Para utilizar esta función se debe establecer previamente una sesión ssh y permite cerrar las conexiones ssh previamente creadas.
- **crear_archivo(n_archivo,v_cargar):** Esta función permite crear un archivo txt en el sistema operativo, para emplearla se debe indicar en las variables: n_archivo el nombre del archivo junto con su ruta y en v_cargar el contenido que se desea llevar al archivo de texto.
- **comando_linux(comando):** Esta función permite ejecutar desde el intérprete Python un comando sobre el sistema operativo Linux, para usarla solo se debe indicar en la variable comando la línea de comando que se desee ejecutar.
- **leer_archivo(archivo):** Esta función permite leer el contenido de un archivo de texto del sistema operativo, esta función retorna sobre la variable mensaje el contenido del archivo leído, para emplearla solo se debe indicar sobre la variable archivo la ruta y el nombre del archivo que se desee leer.
- **exp_text(regexp,texto):** Esta función aplica expresiones regulares sobre un texto, a su vez retorna en la variable result_exp el resultado de la expresión regular, para utilizar esta función sobre las variables: regexp se debe indicar la expresión regular a aplicar y sobre la variable texto la cadena de caracteres a la cual se le quiera aplicar la expresión regular.
- **find_wilcard(csir_mask):** Esta función permite convertir al formato wildcard una máscara que este en formato csir (numérico /24, /8, etc), esta función retorna el valor de la máscara wildcard sobre la variable mask, para usar esta función solo se debe indicar en la variable csir_mask el valor de la máscara sin el “/”.
- **find_mask(csir_mask):** Esta función encuentra el valor de una máscara en formato csirt, dentro de la variable mask retorna el valor de la máscara que se desee encontrar, por ejemplo, para una máscara en /24 retorna el valor 255.55.255.0, para

usar esta función solo se debe indicar en la variable csir_mask el valor de la máscara sin el “/”.

- **contarcaracter(cadena,caracter):** Esta función cuenta caracteres dentro de una cadena de texto, sobre la variable cuenta retorna el número de caracteres que se desee contar en la cadena, por ejemplo si se desea contar el carácter “a” dentro de la cadena “casa”, se retornaría en la variable cuenta el número 2, para usar esta función se debe indicar en la variable cadena el texto y en la variable carácter el carácter que se desee contar de la cadena.
- **modificar_avariables(texto,v_cargar):** Dentro de la arquitectura del SCGI se tiene el archivo de configuración y sincronización de scripts, este archivo se encuentra ubicado dentro de la ruta “/etc/tesis/variablespyton/variables.txt” y esta función permite crear nuevas entradas sobre este archivo bajo el formato establecido (variable#valor), para utilizar esta función es necesario indicar en la variable texto el nombre del campo que se desee crear y en la variable v_cargar el valor del campo.
- **consultar_variable(c_variable):** Esta función permite consultar una variable que se encuentre almacenada en el archivo de configuración y sincronización de scripts, para utilizarla solo es necesario indicar sobre la variable c_variable el campo que se desee consultar y la función retornara sobre la variable “variable” el valor del campo consultado.
- **enviar_log(logger):** Esta función consulta en el archivo de configuración y sincronización de scripts el valor del servidor syslog y envía el contenido de la variable ingresada en logger al servidor.

Las funciones listadas a continuación requieren que previamente se halla establecido una conexión ssh.

- **asegurar_consola():** Esta función aplica el requisito STIG-V-4582 y asegurar la entrada a la línea de consola**.**
- **proteccion_password():** Esta función aplica el requisito STIG-V-3062 y oculta la visualización de password en los archivos de configuración.
- **asegurar_snmp():** Esta función aplica los requisitos STIG-V-3210 y V-3043 para asegurar los servicios e snmp.
- **asegurar_usuradmin(sw):** Esta función aplica los requisitos SITG-V-3143, V-3056 y V-15434 y asegura el ingreso por ssh a los dispositivos de red, adicional la función

consulta el archivo de configuraciones y sincronización de scripts para tomar los datos de autenticación, para hacer uso de la función es necesario indicar sobre la variable sw la ip del dispositivo al que se desea aplicar las buenas prácticas en las credenciales de acceso.

- **asegurar_rometemgmt():** esta función aplica una lista de control de acceso (ACL) en las líneas de gestión remota para accesos vía ssh.
- **asegurar_mgmtvlan(sw):** Esta función aplica el requisito STIG-V-5628 asegurando la vlan de gestión, para utilizar esta función es necesario indicar sobre la variable sw la ip del sw que se desea asegurar.
- **config_ntp():** Esta función aplica el requisito STIG-V-3062 asegurando la configuración del servicio ntp.
- **config_syslog():** Esta función aplica el requisito STIG-V-4584 asegurando la configuración del servicio de eventos syslog.
- **asegurar_ssh():** Esta función aplica los requisitos STIG-V-14717 y V-5613 asegurando la configuración del servicio ssh.
- **asegurar_servicios():** Esta función aplica los requisitos STIG-V-5615, V-3078, V-3079, V-3085 y V-14669 asegurando los diferentes servicios que son ofrecidos sobre los dispositivos de red.

En el anexo 1 – funciones_tesis se entrega el código fuente para el módulo funciones_tesis.py:

Entendiendo un poco mejor la estructura del SCGI y las funciones que lo comprenden se procede a detallar cada una de las fases que permiten alcanzar el objetivo de la tesis.

### 2.2.5 Implementación y desarrollo para la fase de preparación.

Antes que nada, se aclara que sobre esta fase se desarrollan únicamente las entradas y sentencias para dispositivos cisco, sin embargo, la adaptación para otros tipo de fabricantes es relativamente sencilla, pues solo es necesario indicar las funciones de aseguramiento para el fabricante que se desee asegurar, para esta fase se crean dos scripts, uno para lanzar el proceso de recolección de información necesaria para el

aseguramiento y otro para aplicar el archivo de configuración sobre la red de acceso. En la figura 45 se observa la estructura de cada uno de los scripts.

**Script 1 → Ingresar_configuracion.sh**
#!/bin/bash
python3.6 /root/eclipse-workspace/tesisV3/Variables.py

**Script 2 → asegurar_red.sh**
#!/bin/bash
python3.6 /root/eclipse-workspace/tesisV3/Asegurar_red.py

Figura 45. Scripts para la fase de preparación

Se observa las siguientes sentencias en script1 y script2:

#!/bin/bash → Esta sentencia indica que el archivo al ser ejecutado se hará mediante el intérprete bin/bash
Python3.6 → Esta sentencia ejecuta el compilador Python para el módulo de la ruta.
Es decir, cada uno de los scripts se encarga de ejecutar los módulos de variables.py y asegurar_red.py

## 2.2.5.1Variables.py

Variables.py es un módulo en Python que al ser ejecutado genera un conjunto de preguntas al administrador de la red con el fin de recolectar la información necesaria para generar el archivo de configuración y sincronización de scripts.

las preguntas que variables.py realiza son las siguientes:

- ¿Está seguro que quiere iniciar el aseguramiento? Los archivos de configuración serán reescritos s/n:
- ¿Los dispositivos son administrable por CLI (SSH)? s/n:
- ¿Cuál es el usuario administrador actual?:
- ¿Cuál es la contraseña de ingreso actual?
- ¿Cuál es o será la vlan de gestión?
- ¿Cuál debe ser el segmento de administración? (x.x.x.x/x):
- ¿La red tiene un diseño jerárquico? (s/n):
- ¿Cuál es la IP del SW core? (x.x.x.x)

- ¿Cuáles son las ip del sw de distribución? (x.x.x.x,x.x.x.x,end)
- ¿Cuál es la ip del Switch que gestiona las VLAN o el direccionamiento IP? (x.x.x.x)
- ¿La autenticación será por medio de AAA? (s/n)
- Ingrese la dirección IP del AAA (x.x.x.x)
- ¿El servicio es Radius o Tacacs? (radius/tatacs)
- ¿Cuál será el usuario administrador?
- ¿Cuál será la contraseña de administración?
- ¿Cuál es la IP del servidor NTP?
- ¿El servidor NTP tiene autenticación? (s/n)
- ¿Cuál es la autenticación del servicio ntp?
- ¿Cuál es la IP servidor syslog?
- ¿Dispone de recursos para gestionar y mantener un esquema de port-security?
- ¿Los dispositivos de red admiten 802.1x? (s/n)
- ¿El dispositivo es Cisco o Aruba Networks (cisco/aruba)?
- ¿Cuál es la clave del usuario privilegiado?
- ¿Cuál es la comunidad snmp?
- ¿Cuál es la red permitida para indagar snmp (x.x.x.x/x)?
- ¿Cuál es el servidor smtp?
- ¿Cuál es el buzón desde donde se enviarán los correos (From)?
- ¿Cuál será el contacto (correo) de notificación de hallazgos?
- ¿Cuál es la contraseña del buzón de correo de envío?

Cada respuesta a cada pregunta genera una entrada en el archivo de configuración y sincronización de scripts ubicado en la ruta /etc/tesis/variablespyton/variables.txt mediante la función modificar_variable() del módulo funciones_tesis.py, conservando el siguiente formato, “variable#respuesta”. Este formato permite estandarizar la manera en cómo se registra la información de la red permitiendo que otros scripts y módulos hagan uso de la información allí depositada.

Una vez el administrador de red ingresa la información solicitada a través del módulo variables.py se genera el archivo de configuración que antes de su cifrado se ve como en la imagen de la figura 46.

```
root@ubuntu:/etc/tesis/variablespyton# cat variables.txt
#Este archivo guarda las variables para integrar los diferentes modulos


cli#s
Usuario-actual#cesar
Clave-actual#abcd1234
Vlan-gestion#10
wildcard-gestion#0.0.0.3
mascara-gestion#255.255.255.252
red-gestion#10.0.0.0
Modelo-jerarquico#n
sw-core#10.10.1.2
radius#n
conf-user#soporte
Conf-password#Abcd1234*
ntp#10.0.0.2
clave-ntp#n
rsyslog#10.0.0.2
portsecurity#n
autenticacion#n
fabricante#cisco
oid#1.3.6.1.2.1.4.22.1.2
enable_pass#abcd1234
snmpcomunity#lab-tesis
mascara-snmp#0.0.0.255
red-snmp#10.0.0.0
smtp#smtp.gmail.com
buzon#scgi.tesis@gmail.com
notificacion#cesar.rios50545@gmail.com
clave_mail#SCGI50545
configurar#n
asegurado#s
root@ubuntu:/etc/tesis/variablespyton#
```

Figura 46. Archivo de configuración y sincronización de scripts.

Ya terminada la creación del archivo de configuración se procede a su cifrado mediante herramientas GPG (GNU private guard). En la figura 47 se observa la salida del archivo cifrado una vez es desplegado en pantalla.

```
root@ubuntu:/etc/tesis/variablespyton# cat variables.txt.gpg
      bmW◆◆F◆◆◆;◆E◆+◆g7k`Б◆3_o
                                    s◆cń#◆;◆K◆U◆◆◆:Kr#@◆◆$◆◆◆◆◆y
                                                                  L◆◆En◆◆◆x      j◆◆◆N|◆◆!h"◆O◆        <◆j◆
◆f◆◆V◆◆◆◆{◆    ◆◆<◆.◆◆d◆◆.◆&r]◆Qj◆◆◆◆M◆◆◆◆4I◆◆◆◆]◆j◆{loT◆◆Q~h◆◆◆      '◆V◆◆◆e◆◆◆◆◆◆◆◆◆V◆◆◆◆b,◆f◆◆<
                                                                                                  ◆X◆◆◆#eSN*◆◆k◆◆
&◆Z◆◆◆◆◆N◆◆◆*◆F<i◆◆◆◆◆6:◆◆◆◆◆M◆◆◆◆◆◆
◆◆|◆◆d◆◆2◆◆
          ◆}◆iQ`&◆x◆◆,s.◆cI◆◆)◆#◆J◆Ze  ◆◆◆J◆◆◆◆◆o.◆◆6B◆◆◆~◆◆O◆◆%◆a4Z◆◆◆. ◆d◆◆|◆◆P◆◆◆K◆JvOV◆Z◆k◆◆◆|◆◆clfQlc◆S◆◆zzY◆yd
```

Figura 47. Archivo de configuración y sincronización de scripts cifrado.

El anexo 2 – Varibles, entrega el código fuente del módulo variables.py

Una vez creado el archivo de configuración y sincronización de scripts es posible lanzar el script aseguramiento de la red mediante el módulo de Python asegurar_red.py.

## 2.2.5.2Asegurar_red.py

Este quizás sea uno de los módulos más livianos que comprende la arquitectura del SCGI y es dado a que todos los elementos para asegurar la red están sobre los módulos

anteriormente ilustrados, las funciones que aplican los diferentes requisitos del STIG (Security tecnical implementación guide) en el módulo funciones_tesis.py y la información para asegurar la red sobre el archivo de configuración y sincronización de scripts generados desde el módulo variables.py, ya solo restaría aplicar las funciones con la información de configuración sobre la infraestructura de red.

Para aplicar estas configuraciones se utiliza ampliamente la función consultar_variables del módulo funciones_tesis.py, pues esta función permite consultar la información contenida en el archivo de configuración y cargarla en variables locales del módulo asegurar_red.py y así posteriormente ejecutar las demás funciones que aseguran la red. En la figura 48 se observa la manera en cómo se cargan estas variables.

```
"""-----------------------------------------Declar variables-------------------


consultar_variable('sw-core')
sw_core = variable
consultar_variable('sw-distribucion')
swdistribucion = variable
consultar_variable('Modelo-jerarquico')
Modelojerarquico = variable
consultar_variable('fabricante')
fabricante = variable
consultar_variable('Usuario-actual')
usuario = variable
consultar_variable('Clave-actual')
clave = variable
consultar_variable('enable_pass')
enable_password = variable
consultar_variable('rsyslog')
rsyslog = variable
```

**Figura 48. Declaración de variables a partir de la función consultar_variable().**

Se observa el uso de la función consultar_variable, inicialmente se realiza el llamado de la función junto con el campo que se desea consultar, para el primer caso se consulta el campo llamado sw-core que según la figura 46 tendría el valor 10.10.1.2, la función consultar variable, retorna el valor consultado sobre la variable "variable" por ello es que posteriormente se observa la línea sw_core = variable, con el fin de llevar a la variable sw_core el valor 10.10.1.2.

Una vez cargadas todas las variables, solo restaría hacer el llamado de las funciones contenidas en funciones_tesis.py y aplicarlas en un ciclo sobre todos los dispositivos de la red.

En el anexo 3 – asegurar red, se encontrará el código fuente del módulo asegurar_red.py

De esta manera, y a partir de los módulos variables.py y asegurar_red.py se da cumplimiento a la fase de preparación para la atención de incidentes de seguridad relacionados con ataques tipo spoofing, asegurando las configuraciones de los dispositivos de red de manera automática.

### 2.2.6 Implementación y desarrollo para la fase de detección.

Como se expuso en la fase de diseño, se establecen dos métodos de detección, uno desde agentes instalados en estaciones protegidas con sistema operativo Linux para ARP, DNS y DHCP Spoofing y otro directamente desde el SGCI para ARP Spoofing, a continuación, se exponen las técnicas y algoritmos utilizados.

### 2.2.6.1Implementación detección por medio de agentes.

Este método de detección al ser directamente sobre los agentes, exige que exista previamente un esquema de transmisión confiable desde los agentes al SCGI, por ello se iniciara con el contexto del desarrollo de este esquema, en la figura 49 se observa la arquitectura.

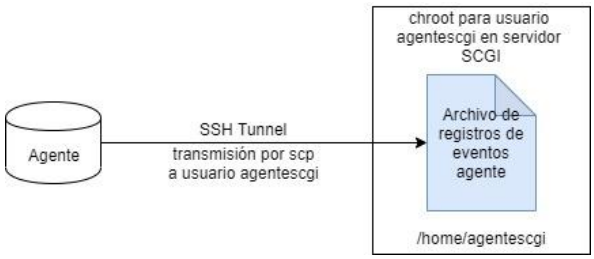


**Figura 49. Arquitectura de comunicación de agentes.**

Como se aprecia en la figura 49 para el desarrollo del método de comunicación se utilizó un túnel de conexión ssh el cual por medio de cifrado asimétrico comparte una llave compartida aleatoria con la cual se establece el túnel de conexión seguro, este método de comunicación es ampliamente utilizado por protocolos de transferencia de archivos en ambientes productivos, un claro ejemplo son los escenarios de SFTP (SSH File transport protocol) y SCP (secure copy) [44].

Para la implementación de la figura 49 se creó el usuario local “agentescgi” en el servidor SCGI con directorio raíz /home/agentescgi, adicional y con el fin de asegurar el acceso remoto del usuario se crea una jaula chroot (change root) sobre este directorio con el fin de evitar, que quien se autentique con este usuario pueda salir de dicho directorio raíz y navegar por los diferentes directorios del sistema operativo, además, sobre el archivo de configuración /etc/ssh/sshd_config se permite el acceso ssh para el usuario agentescgi y en el archivo de sudoers /etc/sudoers (super user do) se crea una regla de acceso para que dicho usuario solo pueda ejecutar acciones de copy (agentescgi ALL = /bin/cp).

De esta manera finamente se crea en la ruta /home/agentescgi un archivo plano llamado notificaciones donde los agentes registraran los diferentes eventos que se vallan detectando, para que así posteriormente el SCGI pueda realizar las acciones de contención correspondientes.

Con todo preparado por parte del SCGI para recibir los diferentes eventos solo resta acondicionar y preparar los agentes para que estén en la capacidad de detectar y enviar los hallazgos al SCGI, para ello se establece la arquitectura de la figura 50.

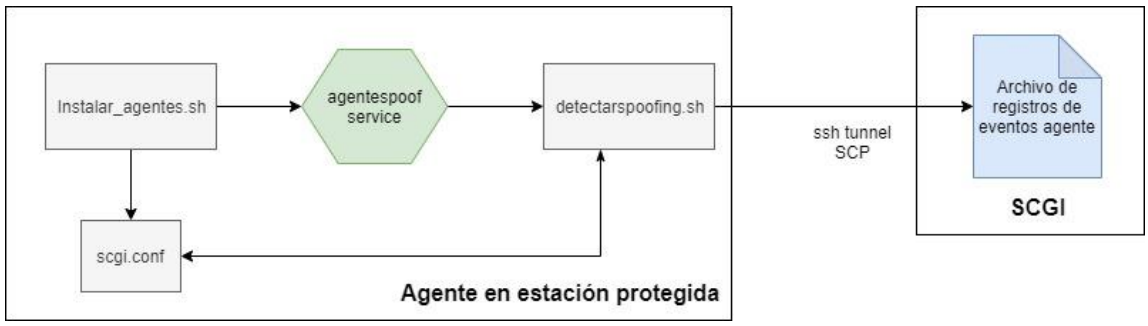


**Figura 50. Arquitectura para los agentes.**

Como se observa en la figura 50 desde el script instalar_agentes.sh se acondicionan los diferentes elementos que permiten realizar la detección y envío de eventos tipo spoofig, este script crea el archivo de configuración scgi.conf, el servicio agentespoof.service y el script detectarspoof.sh los cuales se explican a continuación.

**Scgi.config:** este archivo se encarga de registrar la información correcta de la red, esto con el fin de servir como referencia de comparación para identificar posibles anomalías en la red, la creación de este archivo se realiza por medio de las siguientes preguntas que deben ser contestadas por el administrador de la red:

- Por favor ingrese la IP del SCGI
- Para una correcta detección en la suplantación del default-gateway se debe ingresar la IP pública del SCGI.
- ¿Cuál es la puerta de enlace para la red?
- ¿Cuántos servidores DNS autorizados tienen en la red?
- Digite los DNS

Una vez ingresadas las respuestas el archivo de configuración se ve como se ilustra en la imagen 51.

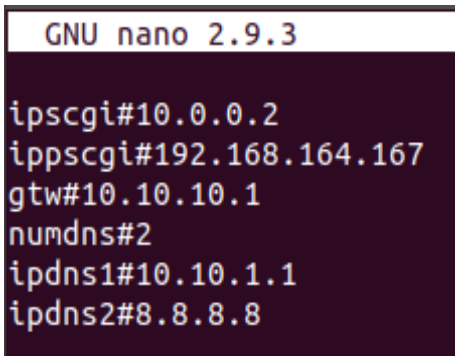


Figura 51. Scgi.config en agentes.

**agentespoof.service:** Este servicio se encarga de ejecutar cada vez que el sistema operativo inicie el script creado sobre la ruta /etc/SCGI/detectarspoofing.sh, script que contiene todos los métodos de detección y esquemas de envió de hallazgos al SCGI.

**Detectarspoof.sh:** detectarspoof es el script que integra todos los métodos de detección para ARP Spoofing, DNS Spoofing y DHCP Spoofing, junto con los métodos para el envío de los hallazgos al SCGI, para ello inicialmente se deben cargar los datos necesarios desde el archivo scgi.config e indagar los parámetros IP que identifiquen la estación protegida en el SCGI una vez se realice la notificación., en la figura 52 se observa las líneas de comandos usadas para dicho fin.

```
#!/bin/bash

#reiniciar los servicios de diagnostico de parametros IP propietario de linux
service NetworkManager restart

sleep 2

GWSPOOF=0
APRSPOOF=0
DNSSPOOF=0

while true;do

#identificar el HOST para el envio de las notificaciones al SCGI
IPSCGI=$(cat /etc/SCGI/scgi.config | grep ipscgi | grep -o [0-9.].*)
IPHOST=$(nmcli dev sho | grep -o IP4.ADDRESS.*/ | head -n 1 | sed 's/IP4.ADDRESS.*://g' | sed 's/ //g' | sed 's/\///g')
MACHOST=$(nmcli dev sho | grep GENERAL.HWADDR.* | head -n 1 | sed 's/GENERAL.HWADDR://g' | sed 's/ //g')
```

**Figura 52. Configuración previa para iniciar con detección y notificación desde agentes.**

Se observa que inicialmente se indica el intérprete con el cual debe ser ejecutado el archvio.sh /bin/bash, seguidamente se reinician los servicios de gestión de redes con el fin de restablecer cualquier dificultad o error cargado en el mismo una vez inicie el sistema operativo, posteriormente se cargan las variables GWSPOOF, ARPSPOOF y DNSSPOOF en 0, esto con el fin de iniciar el servicio sin hallazgos ni detecciones y finalmente se tienen las sentencias que cargan la información para él envió de las notificaciones:

- **IPSCGI=$(cat /etc/SCGI/scgi.config | grep ipscgi | grep -o [0-9**.].*): Esta sentencia aplica una expresión regular con grep a la salida del despliegue del archivo scgi.conf buscando la coincidencia “ipscgi” posteriormente aplica una nueva expresión regular “[0-9.].*” indicando únicamente la IP del SCGI y la carga en la variable IPSCGI.

- **IPHOST=$(nmcli dev sho | grep -o IP4.ADDRESS.*/ | head -n 1 | sed 's/IP4.ADDRESS.*://g' | sed 's/ //g' | sed 's/\///g'):** Esta sentencia utiliza la herramienta nmcli (network manager command line), básicamente lo que permite es que a nivel de línea de comandos (CLI) se pueda controlar o manipular la gestión de la red [45], para este caso se despliega la información de las diferentes tarjetas de red con el fin de buscar la dirección IPv4 configurada en la interface principal o que da salida a la red.

- **MACHOST=$(nmcli dev sho | grep GENERAL.HWADDR.* | head -n 1 | sed 's/GENERAL.HWADDR://g' | sed 's/ //g'):** Nuevamente se hace uso de la herramienta nmcli, pero en esta ocasión en con el propósito de conocer la dirección MAC de la tarjeta de red que da salida a la red.

Estas tres sentencias son de gran importancia dentro del script pues son las que cargan la información que permite posteriormente identificar la estación comprometida en el evento que se envía al SCGI.

Ya preparado el terreno para la notificación de los eventos es posible adentrarse en los métodos de detección disponibles sobre el script detectarspoof.sh.

**Detección de ARP Spoofing:** Para este módulo se emplea la sentencia “arp -a”, la salida de este comando despliega la tabla ARP la cual permite revisar si existen MAC duplicadas para una misma IP y en caso de encontrar una entrada duplicada se activa el procedimiento de notificación, en la imagen 53 se muestran las líneas de código que permiten materializar este concepto.

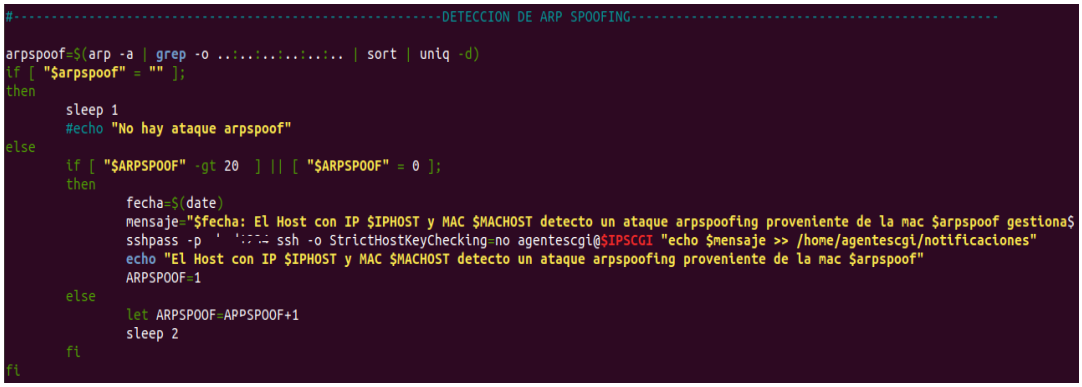


Figura 53. Código para la detección de ARPSpoof desde los agentes.

- **arpspoof=$(arp -a | grep -o ..:..:..:..:..:.. | sort | uniq -d):** Esta línea de comando carga en la variable arpspoof la salida de la expresión regular (grep -o ..:..:..:..:..:..) quien busca las direcciones MAC de la salida del comando arp -a, posteriormente organiza alfabéticamente la salida con el comando sort y finalmente con el comando uniq -d deja en memoria únicamente las entradas repetidas.
- **if [ "$arpspoof" = "" ]:** Esta línea de comandos aplica la comparación lógica if (si) indagando si la variable arpspoof se encuentra vacía, pues esta siempre debe estar en estado vacío a menos que exista alguna detección, por ello se observa en las líneas posteriores de la figura 52 que si la variable se encuentra vacía (then) el condigo no hace nada, pero en el caso contrario (else) se ejecuta el proceso de notificación.
- **Líneas del proceso de notificación:** En las primeras líneas del proceso de notificación se observa el condicional if [ "$ARPSPOOF" -gt 20 ] || [ "$ARPSPOOF" = 0 ]; que quiere decir, si la variable ARPSPOOF es mayor de 20 o igual a cero ejecuta las sentencias incluidas en el Then siguiente, esto es con el fin de que el

servicio de detección no dispare notificaciones masivas una vez encuentra un hallazgo, en su lugar lo hace cada 40 seg pues en el caso donde no se cumpla la condición (else) se incrementa el contador de ARPSPOOF y aguarda 2 segundos (2 segundos * 20 = 40 segundos), por otro lado si el algoritmo entra en el proceso de notificación se crea inicialmente el mensaje de alerta concatenando la fecha del hallazgo, el tipo de ataque, Ip comprometida, mac comprometida, mac del atacante y un campo al final de gestionar para que el SCGI sepa que tienen un evento por gestionar. Finalmente se establece el túnel ssh y se transmite el evento al SCGI por medio de SCP en la línea “sshpass -p ****** ssh -o StrictHostKeyChecking=no agentescgi@$IPSCGI "echo $mensaje >> /home/agentescgi/notificaciones"

**Detección de DNS Spoofing:** este método de detección se basa únicamente en las salidas del comando nmcli y la comparación de los registros contenidos en el archivo scgi.config, en la figura 54 se observan las líneas de comandos que integran este módulo de detección.

```
#-----------------------------------------------------DETECCION DE FALCIFICACION DE DNS--------------------------------------

#sacar el numero de DNS que se deben revisar
numdns=$(cat /etc/SCGI/scgi.config | grep numdns# | sed 's/numdns#//g')

#sacar los DNS configurados en el OS
CONFDNS=$(nmcli dev sho | egrep DNS | sed 's/IP4.DNS.*://g' | sed 's/ //g')

reemplazo=$(echo $CONFDNS | tr ' ' ,)

#generar array con scripts a aplicar
IFS=', ' read -a ascript <<< "$reemplazo"

#Verificar si los DNS autorizados son los correspondientes a los configurados
for IPDNS in "${ascript[@]}"
do
        FIND=$(cat /etc/SCGI/scgi.config | grep -o ip.*#$IPDNS)
        if [ -s $FIND ]
        then
                if [ "$DNSSPOOF" -gt 20  ] || [ "$DNSSPOOF" = 0 ];
                then
                        fecha=$(date)
                        macatack=$(arp -a | grep $IPDNS | grep -o at.*:..:..:..:..:.. | sed 's/at //g')
                        mensaje="$fecha: El Host con IP $IPHOST y MAC $MACHOST detecto que el DNS $IPDNS con MAC $macatack no se encuentra au$
                        sshpass -p ******  ssh -o StrictHostKeyChecking=no agentescgi@$IPSCGI "echo $mensaje >> /home/agentescgi/notificacio$
                        echo $mensaje
                        #echo "El Host con IP $IPHOST y MAC $MACHOST detecto que el DNS $IPDNS con MAC $macatack no se encuentra autorizado p$
                        DNSSPOOF=1
                else
                        let DNSSPOOF=DNSSPOOF+1
                        sleep 2
                fi
```

Figura 54. Código para la detección de DNS Spoof desde los agentes.

- **numdns=$(cat /etc/SCGI/scgi.config | grep numdns# | sed 's/numdns#//g'):** Esta línea carga sobre la variable numdns el número de DNS autorizados sobre la red indicados sobre el archivo de configuración.

- **CONFDNS=$(nmcli dev sho | egrep DNS | sed 's/IP4.DNS.*://g' | sed 's/ //g'):** Esta línea de comando carga sobre la variable CONFDNS los dns configurados actualmente en la estación protegida, esto por medio de expresiones regulares y la aplicación de la herramienta nmcli.
- Las dos líneas de código que se observan a continuación buscan convertir la variable CONFDNS en un array de tantas posiciones como DNS existan configurados, esto es con el fin de comparar cada DNS configurado sobre un ciclo for con los DNS autorizados.
- **FIND=$(cat /etc/SCGI/scgi.config | grep -o ip.*#$IPDNS):** Esta línea es la encargada de buscar si los DNS configurados en la maquina protegida se encuentran en la lista de DNS autorizados sobre el archivo scgi.config, la indagación se realiza por medio de una expresión regular buscando una coincidencia de ambos DNS, en el caso de que no exista un DNS Spoofing la variable FIND quedara cargada con la información del DNS consultado, pues la expresión regular "grep" habrá encontrado una coincidencia entre ambos DNS, en caso contrario la variable FIND estará vacía pues no existirá una coincidencia entre ambos DNS.
- **if [ -s $FIND ]:** Esta función comparativa if con la condición -s indaga si la variable FIND se encuentra vacía o no, en caso de no estar vacía ejecuta las sentencias de notificación ubicadas después del primer then**,** en caso contrario el servicio de detección no hará nada.
- El proceso de notificación es exactamente igual al explicado sobre el módulo de detección de ARPSpoofing sin embargo difiere en la siguiente línea.
- **macatack=$(arp -a | grep $IPDNS | grep -o at.*:..:..:..:..:.. | sed 's/at //g'):** Esta línea tiene como propósito identificar sobre la tabla ARP la MAC del DNS suplantado, esto con el fin de enviarla en la notificación para que el SCGI pueda aislarla posteriormente en el proceso de contención.

**Detección de DHCP Spoofing:** Este módulo de detección tiene algunas consideraciones particulares, pues dentro de su materialización el ataque generalmente suplanta el default Gateway creando su propia red e interceptando el tráfico que la víctima generé hacia redes externas, por lo que es muy probable que se aislé la victima de la red local en donde se encuentra el SCGI.

En la figura 55 se observa un posible escenario en donde la victima queda totalmente aislada de la red local con el SCGI, por lo que es necesario establecer un nuevo camino vía internet para que las notificaciones y alertas lleguen al SCGI y si bien es cierto que el atacante podría de otra manera materializar el ataque creando un nuevo default Gateway y sobre otra interface de red retornar el tráfico a la red interna para que la víctima no sospeche que está siendo interceptada, es necesario que la técnica de detección cubra los diferentes escenarios y logre enviar las notificaciones el SCGI.

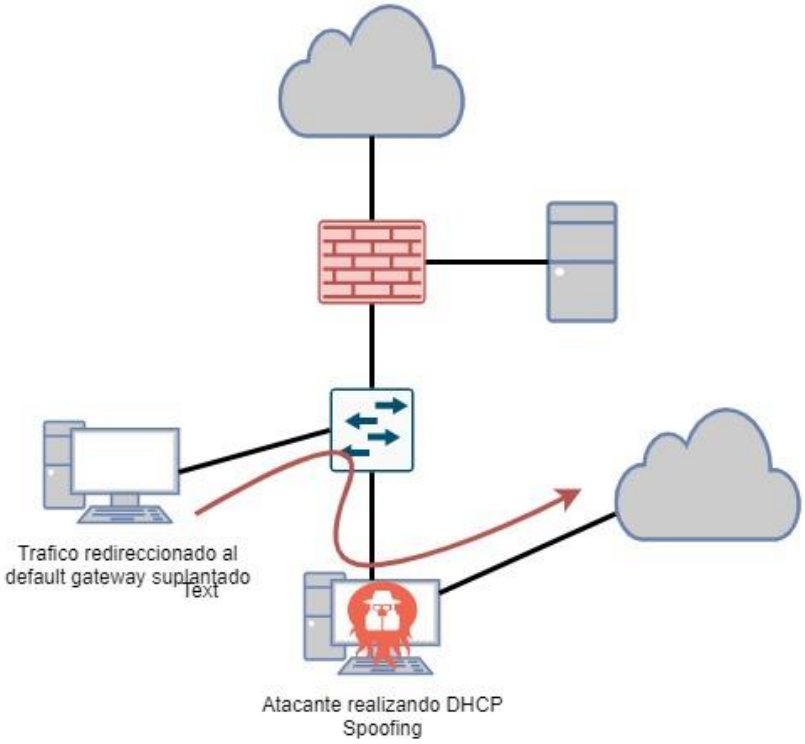


**Figura 55. Posible escenario de DHCP Spoofing.**

Una vez aclarado esto, se explicará el código del módulo de detección de DHCP Spoofing ilustrado en la figura 56.

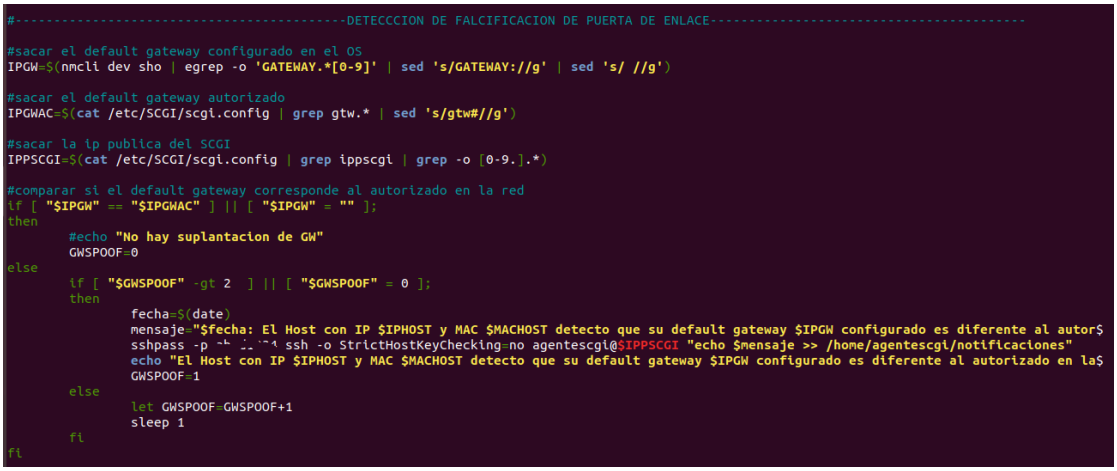


**Figura 56. Código para la detección de DHCPSpoofing desde los agentes**

- **IPGW=$(nmcli dev sho | egrep -o 'GATEWAY.*[0-9]' | sed 's/GATEWAY://g' | sed 's/ //g'):** Al igual que el módulo de detección de DNS Spoofig, este módulo utiliza la herramienta nmcli para identificar el default Gateway configurado en la actualidad sobre la estación protegida, mediante la expresión regular egrep -o 'GATEWAY.*[0-9]' se busca la ip del default Gateway y con los comandos siguientes (sed 's/GATEWAY://g' | sed 's/ //g')) se limpia la salida dejando únicamente la IP para ser cargada sobre la variable IPGW
- **IPGWAC=$(cat /etc/SCGI/scgi.config | grep gtw.* | sed 's/gtw#//g'):** Esta sentencia carga sobre la variable IPGWAC el default Gateway autorizado sobre la red presente en el archivo de scgi.config.
- **IPPSCGI=$(cat /etc/SCGI/scgi.config | grep ippscgi | grep -o [0-9.].*):** Esta sentencia carga sobre la variable IPPSCGI la ip publica en donde se tiene publicado el SCGI, esto con el fin de enviar las notificaciones y alertas vía internet.
- **if [ "$IPGW" == "$IPGWAC" ] || [ "$IPGW" = "" ]:** Este condicional IF tiene como propósito identificar si el default Gateway fue suplantado a través de un DHCP Spoofing, comparando el Default Gateway actual vs el autorizado, adicional revisa si el valor del Default Gateway actual es vacío, ya que es muy posible que al iniciar el sistema operativo este aun no haya sido asignado por un servidor DHCP. En el caso de cumplirse la condición se entiende que no hay un DHCP Spoofing, por lo que el algoritmo no hará nada en la sentencia “then”, pero si el condicional da como

resultado una negación se activará el proceso de notificación bajo las líneas indicadas en la sentencia “else”.

- El proceso de notificación es exactamente igual al explicado sobre el módulo de detección de ARPSpoofing con la única diferencia que la notificación será lanzada a la IP pública del SCGI indicada sobre la variable IPPSCGI, esto se aprecia en la línea “sshpass -p **** ssh -o StrictHostKeyChecking=no agentescgi@$IPPSCGI "echo $mensaje >> /home/agentescgi/notificaciones"

Finalmente, el anexo 4 – instalar agente, entrega el código fuente completo del agente para su instalación. (instalar_agente.sh).

### 2.2.6.2 Implementación para la detección de ARP Spoofing directamente desde el SCGI

Este método de detección se aplica directamente sobre la red de acceso ampliando la cobertura de protección sobre los dispositivos pertenecientes a la red, ya que no depende de la instalación de agentes en las diferentes estaciones, en la figura 57 se observa la arquitectura de este servicio.

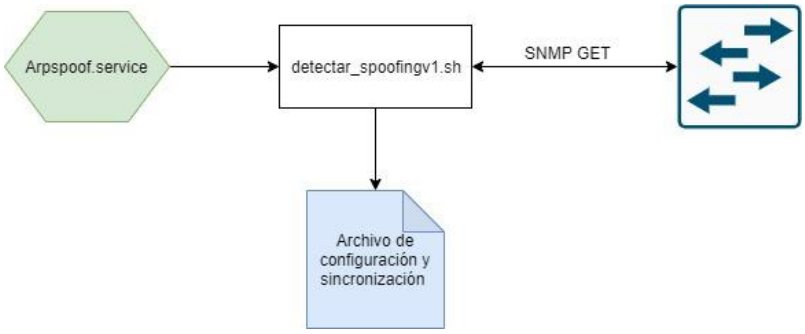


**Figura 57. Arquitectura de detección de ARP Spoofing directamente desde el SCGI.**

La arquitectura ilustrada en la figura 57 es bastante simple, ya que solo es necesario una conexión snmp al dispositivo que gestiona el direccionamiento IP de la red, desde allí se realiza la investigación de la tabla arp, a continuación, se detalla cada uno de los componentes de la arquitectura.

**Arpspoof.service:** este servicio arranca cuando el sistema operativo inicia y su único propósito es ejecutar el script detectar_spoofingV1.sh, este servicio se encuentra ubicado sobre la ruta /etc/systemd/system/agentespoof.service y en la figura se muestra su estructura.

```
[Unit]
After=network.target
After=systemd-user-sessions.service
After=network-online.target

[Service]
User=root
ExecStart=/bin/bash /etc/tesis/detectar_spoofingv1.sh

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

Se observa como desde la línea ExecStart se ejecuta el script ubicado en la ruta /etc/tesis/detectar_spoofingv1.sh bajo el intérprete /bin/bash.

**Detectar_spoofingv1.sh:** Como se ha mencionado anteriormente este script se encarga de realizar la detección del ARP Spoofing directamente sobre la red indagando constantemente la tabla ARP por medio de SNMP, para ello inicialmente se cargan las variables para la consulta SNMP desde el archivo de configuración y sincronización de scripts ubicado en la ruta /etc/tesis/variablespyton/variables.txt y después se realiza la consulta snmp por medio de la herramienta snmpwalk disponible desde Linux. En la figura 58 se observa el proceso anteriormente descrito.

```
#!/bin/bash
arpoid=$(grep 'oid.*' /etc/tesis/variablespyton/variables.txt | sed 's/oid#//g')
comunity=$(grep 'snmpcomunity.*' /etc/tesis/variablespyton/variables.txt | sed 's/snmpcomunity#//g')
ipgw=$(grep 'sw-core.*' /etc/tesis/variablespyton/variables.txt | sed 's/sw-core#//g')
#echo $comunity $ipgw $arpoid
#sleep 60
while true;do
snmpwalk -v2c -c $comunity $ipgw $arpoid > show_arp
sed -i 's/A/a/g' show_arp
sed -i 's/B/b/g' show_arp
sed -i 's/C/c/g' show_arp
sed -i 's/D/d/g' show_arp
sed -i 's/E/e/g' show_arp
sed -i 's/F/f/g' show_arp
sed -i 's/.$//g' show_arp
sed -i 's/ /:/g' show_arp
```

Figura 58. Ejecución SNMP desde detectar_spoofingv1.sh

En la imagen previa se muestra como la salida de la consulta snmp se guarda en el fichero show_arp bajo el comando “>” y posteriormente con la herramienta sed se intercambian las letras mayúsculas a minúsculas para las diferentes MAC, de esta manera se logra tener

en un archivo la información de la tabla ARP del defualt Gateway de la red y aplicar métodos similares a los expuestos anteriormente en el módulo de detección de agentes para ARP Spoofing. Ver figura 59.

```
macspoof=$(cat show_mac | sort | uniq -d)

if [ -s $macspoof ]
then
        sleep 10
        #echo "no se a detectado un arp spoofing"
else
        IPS=$(grep $macspoof show_arp | grep -o [0-9]*.[0-9]*.[0-9]*.[0-9]*:= | sed "s/.=//g")
        #echo $IPS
        echo "las IPs $IPS con mac $macspoof posiblemente son victimas de un arp spoofing" > mensaje.txt
        sed -i ':a;N;$!ba;s/\n/ /g' mensaje.txt
        sed -i 's/  / /g' mensaje.txt
        cat mensaje.txt
        echo $macspoof > /etc/tesis/variablespyton/macspoof.txt
        cat /etc/tesis/variablespyton/macspoof.txt | sed 's/://g' | grep -o '....$' > /etc/tesis/variablespyton/macspoofcisco.txt
        python3.6 /root/eclipse-workspace/tesisV3/contencion_arp.py
        sleep 10
fi
rm show_arp
rm show_mac
done
```

**Figura 59. Detección de arp spoofing desde detectar_spoofingv1.sh**

En efecto se observa que el método de detección es muy similar al expuesto en el módulo para agentes, sobre la línea “macspoof=$(cat show_mac | sort | uniq -d)” se organizan alfabéticamente y se dejan únicamente las entradas repetidas, así en la línea siguiente “if [ -s $macspoof ]” es posible determinar si se encontraron MAC duplicadas o no, en el caso de la variable macspoof este vacía “then” no se habrá detectado un ataque de arp spoofing y el algoritmo no ejecutaría ninguna acción, pero en el caso contrario “else” donde la variable macspoof no se encuentre vacía indicara que ha detectado un ataque y se deberá ejecutar las siguientes líneas:

- **IPS=$(grep $macspoof show_arp | grep -o [0-9]*.[0-9]*.[0-9]*.[0-9]*:= | sed "s/.=//g"):** carga en la variable IPS las ips comprometidas dentro del ataque de arp spoofing, esto se logra aplicando una la expresión regular “grep $macspoof show_arp | [0-9]*.[0-9]*.[0-9]*.[0-9]*:=” sobre el archivo show_arp
- **cat /etc/tesis/variablespyton/macspoof.txt | sed 's/://g' | grep -o '....$' > /etc/tesis/variablespyton/macspoofcisco.txt:** Esta línea se encarga de crear el archivo de integración con el cual el módulo de contencion_arp.py logra tomar la mac en formato cisco para aislarla de la red.

De esta manera se logra implementar un método de detección sin dependencia de agentes para la amenaza ARP Spoofing, el código fuente completo del script se entrega en el anexo 5 detectar spoofingv1

Una vez abordado los diferentes métodos, módulos, scripts y algoritmos implementados para la detección de amenazas internas tipo spoofing ARP, DNS, DHCP, se dará paso a la implementación de los esquemas de contención y atención del incidente de seguridad detectado.

### 2.2.7 Implementación y desarrollo de la fase de contención.

El propósito de esta fase es aislar la amenaza una vez esta sea detectada y así automatizar la atención del incidente de seguridad, bajo este contexto parecería ser la fase más importante del proceso, sin embargo, es las más sencilla de implementar pues ya todo está dado para su materialización, se tienen las funciones en Python que interactúan con la red de datos, los archivos de configuración con los datos de acceso y la información de la identidad que provoca el incidente de seguridad, solo faltaría relacionar estos datos y lanzar las tareas apropiadas una vez sean necesarias. En la figura 44 se observaban dos métodos sobre los cuales estas tareas son invocadas, una para apoyar las labores de los agentes mediante el servicio agentespoof.service y la otra mediante el script de detección_spoofingv1.sh

#### 2.2.7.1Implementación contención desde los agentes.

Este módulo debe tener visibilidad constante del archivo donde los agentes informan los hallazgos detectados con el fin de actuar frente a una nueva materialización del ataque, para ello cada registro es enviado desde los agentes con la marca de “gestionar” así el servicio de agentes.service sabe cuándo tiene un nuevo evento sin gestionar y así una vez el servicio de contención sea aplicado retira la marca dejando el registro como gestionado. En la imagen 60 se observa la arquitectura de contención para este método.

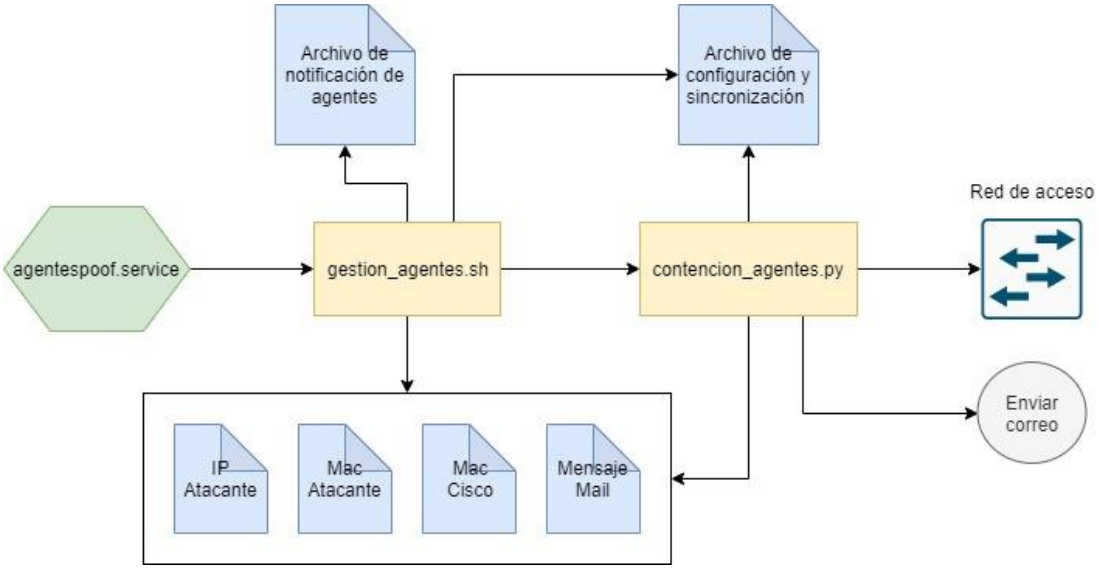


**Figura 60. Arquitectura para la contención de eventos reportados desde los agentes.**

**Agentespoof.service:** este servicio arranca cuando el sistema operativo inicia y su único propósito es ejecutar el script gestion_agentes.sh, este servicio se encuentra ubicado sobre la ruta /etc/systemd/system/agentespoof.service y en la figura 61 se muestra su estructura.

```
[Unit]
After=network.target
After=systemd-user-sessions.service
After=network-online.target

[Service]
User=root
ExecStart=/bin/bash /home/agentescgi/gestion_agentes.sh

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

**Figura 61. agentespoof.service**

Se observa como desde la línea ExecStart se ejecuta con el intérprete /bin/bash el script ubicado en la ruta /home/agentescgi/gestion_agentes.sh

**gestion_agentes.sh:** Este es el script encargado de monitorear el archivo de notificaciones donde los agentes registran sus hallazgos y al momento de encontrar un nuevo incidente acondiciona el escenario para lanzar las acciones de contención desde el

módulo en Python contencion_agentes.py. Sobre la imagen 60 se observa que el script lee el archivo de notificaciones y crea los elementos: IP atacante, MAC atacante, Mac cisco y mensaje Mail; en ficheros del sistema operativo para que el módulo contencion_agentes.py pueda consultarlos al momento de ser lanzado, a continuación, se explicaran las líneas de código que se encargan de materializar este concepto.

```
#!/bin/bash
while true;do
#Identificar si hay eventos nuevos para notificar
evento=$(grep -n gestionar /home/agentescgi/notificaciones | head -1 | grep -o [A-Za-z].* | sed 's/gestionar//g')
numeroevento=$(grep -n gestionar /home/agentescgi/notificaciones | head -1 | grep -o ^[0-9]*)

#Verificar si la red es gestionanble para contener la amenaza
verificar=$(cat /etc/tesis/variablespyton/variables.txt | grep cli | sed 's/cli#//g')
```

**Figura 62. Monitoreo de archivo de notificaciones**

La figura 62 muestra las primeras líneas del script gestion_agentes.sh, inicialmente se establece un ciclo continuo con el “while true;do” esto con el propósito de que una vez el script sea ejecutado desde agestespoof.service este continúe su ejecución revisando el archivo de notificaciones constantemente.

- **evento=$(grep -n gestionar /home/agentescgi/notificaciones | head -1 | grep -o [A-Za-z].* | sed 's/gestionar//g'):** Esta línea es la encargada de evidenciar eventos reportados desde los agentes sin gestionar, se observa la aplicación de la expresión regular “grep -n gestionar**”** que busca la coincidencia de la palabra gestionar sobre el archivo ubicado en la ruta /home/agentescgi/notificaciones, la siguiente sentencia “head -1” deja únicamente el primer evento con el fin de gestionar los registros sobre su orden de llegada y así finalmente con la sentencia sed retirar la marca gestionar para cargar sobre la variable “evento” el evento que debe ser gestionado.
- **verificar=$(cat /etc/tesis/variablespyton/variables.txt | grep cli | sed 's/cli#//g'):** Esta sentencia tienen el único propósito de consultar sobre el archivo de configuración y sincronización de scripts si la red es gestionable por CLI o no, esto con el fin de identificar si es posible aislar la amenaza o no.

Con estas sentencias el script gestion_agentes.sh está en capacidad de identificar oportunamente cuando desde un agente se notifica un nuevo evento, solo restaría explicar

las líneas de código que preparan el escenario de contención y lanza el módulo de Python contencion_agentes.py.

```
if [ "$evento" != "" ];
then
        echo "el evento es $evento"
        echo "el numero del evento es $numeroevento"
        mensaje="En la fecha $evento"
        #sacar la ip del atacante
        IP=$(echo $mensaje | grep -o IP.[0-9]*.[0-9]*.[0-9]*.[0-9]* | sed 's/IP //g')
        #sacar la mac del atacante
        MAC=$(echo $mensaje | grep -o con.MAC...:..:..:..:..:.. | sed 's/con.MAC //g')
        echo $mensaje > /home/agentescgi/mensaje
        cat /home/agentescgi/mensaje
        echo $IP > /home/agentescgi/IP
        echo $MAC > /home/agentescgi/MAC
        #Modificar la MAC para realizar la contencion de la amenaza formato cisco
        sed -i 's/A/a/g' /home/agentescgi/MAC
        sed -i 's/B/b/g' /home/agentescgi/MAC
        sed -i 's/C/c/g' /home/agentescgi/MAC
        sed -i 's/D/d/g' /home/agentescgi/MAC
        sed -i 's/E/e/g' /home/agentescgi/MAC
        sed -i 's/F/f/g' /home/agentescgi/MAC
        sed -i 's/:/:/g' /home/agentescgi/MAC
        #cat /home/agentescgi/MAC
        MACCISCO=$(cat /home/agentescgi/MAC | grep -o ....$)
        echo $MACCISCO > /home/agentescgi/MACCISCO
        #echo "la mac cisco es $MACCISCO"
```

Figura 63. Contención de amenazas desde los agentes.

En la figura 63 se observa que en la primera línea está el condicional “if [ "$evento" != "" ];” que básicamente indaga si existe un nuevo evento por gestionar, ya que si no se han reportado nuevos eventos la variable “evento” estaría vacía por lo explicado anteriormente y bajo el caso que existan eventos por gestionar se ejecutarían las sentencias incluidas dentro del “then”.

- **mensaje="En la fecha $evento":** Carga sobre la variable “mensaje” el evento que debes ser gestionado
- **IP=$(echo $mensaje | grep -o IP.[0-9]*.[0-9]*.[0-9]*.[0-9]* | sed 's/IP //g'):** Carga sobre la variable “IP” la ip del atacante realizando la expresión regular “grep -o IP.[0-9]*.[0-9]*.[0-9]*.[0-9]* | sed 's/IP //g’” sobre la variable evento.
- **MAC=$(echo $mensaje | grep -o con.MAC...:..:..:..:..:.. | sed 's/con.MAC //g'):** Carga sobre la variable “MAC” la mac del atacante.
- **echo $mensaje > /home/agentescgi/mensaje:** Crea el fichero mensaje en la ruta /home/agentes con el mensaje a enviar.
- **echo $IP > /home/agentescgi/IP:** Crea el fichero IP en la ruta /home/agentes con la ip del atacante

- **echo $MAC > /home/agentescgi/MAC:** Crea el fichero MAC en la ruta /home/agentes con la mac del atacante
- Los comandos sed -i convierten las letras mayúsculas de la MAC en letras minúsculas, esto con el fin de acondicionar la mac a formato cisco.
- **MACCISCO=$(cat /home/agentescgi/MAC | grep -o ....$):** Esta sentencia carga sobre la variable “MACCISCO” la mac de atacante en formato cisco.
- **echo $MACCISCO > /home/agentescgi/MACCISCO:** Crea el fichero MACCISCO en la ruta /home/agentes con la mac en formato cisco del atacante.
- **if [ "$verificar" == "s" ];** Este condicional verifica el estado de la variable “verificar” quien conoce si la red es gestionable por CLI o no, en caso de ser gestionable lanza el módulo de Python contencion_agentes.py.
- **sed -i "$ultimosed/ gestionar//g" /home/agentescgi/notificaciones:** Esta sentencia quita la marca de gestionar del evento que acaba de ser gestionado.

De esta manera es que el script gestion_agentes.sh logra identificar nuevos ventos y acondicionar los ficheros para que el módulo de Python contencion_agentes.py pueda realizar su tarea.

**Contencion_agentes.py:** Este es el módulo encargado de interactuar con la red de datos para buscar la amenaza reportada desde su dirección MAC y aislar la estación poniendo en estado shutdown la interface sobre la cual se encuentre la dirección MAC, seguidamente envía el correo de notificación al administrador informando de las acciones realizadas.

Para lograr la misión del módulo en su arquitectura se definen 4 fases:

1. Definición de funciones
2. Declaración de variables
3. Acciones de contención
4. Envió de notificación

**Definición de funciones:** El módulo de contecion_agentes.py hace uso de las siguientes funciones del módulo funciones_tesys.py:

- comando_linux

- cerrar_conn
- crear_archivo
- enviar_log
- exp_text
- consultar_variable
- enviar_correo
- establecer_ssh
- enable
- ejecutar_cmd
- exp_cmd
- leer_archivo

**Definición de variables:** Para interactuar con la red de datos y aislar la amenaza es necesario que el módulo contencion_agentes.py conozca los datos necesarios desde el archivo de configuración y sincronización generado desde el módulo variables.py en la fase de preparación, estas son las variables que el módulo carga:

- mail_password
- msgFrom
- msgTo
- fabricante
- IP sw-core
- IPs sw-distribucion
- enable_pass
- buzón
- smtp
- conf-user
- conf-password
- MACCISCO
- Mensaje para envió de notificación.

**Acciones de contención:** ya con las funciones y los datos de acceso a la red restaría ejecutar los comandos de búsqueda de Mac en las interfaces, apagado de puertos y

borrado de tablas arp, adicional, la aplicación de expresiones regulares que interpreten las salidas de los comandos para dar una secuencia lógica a la ejecución de los comandos, los comandos y expresiones regulares utilizados para esta fase son:

- show mac-address-table | inc MACCISCO
- Aplicar la expresión regular [aA-zZ,0-9]*\/[0-9]*\/*' a la salida del comando anterior, esta expresión entrega el puerto sobre el cual se encuentra la mac.
- configure terminal
- interface “salida de la expresión regular”
- shutdown
- exit
- clear arp

**Envió de notificación:** Esta fase se encarga de notificar al administrador sobre las acciones realizadas en la fase de detección y contención, para ello envía la notificación vía correo electrónico y vía log hacia el servidor de eventos configurado en la red.

En el anexo 6, se entrega el código fuente del módulo contencion_agentes.py

### 2.2.7.2 Implementación contención para detección directamente del SCGI.

La fase de contención para este método es ejecutada desde el módulo contencion_arp.py y que es lanzado directamente desde el script detectar_spoofingv1.sh, en la figura 64 se observa la arquitectura de operación para este método.

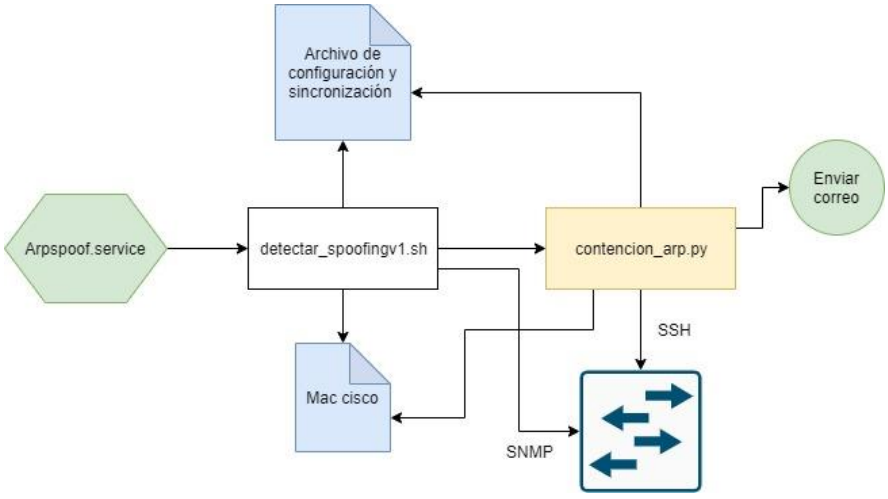


**Figura 64. Arquitectura contención desde detección directa del SCGI.**

La arquitectura ilustrada sobre la figura 63 no difiere mucho de la arquitectura ilustrada previamente sobre la figura 56, y esto es dado a que el método de detección de arp spoofing directamente desde el SCGI prepara todo el escenario para realizar la posterior contención de la amanezca, es tanto que desde el mismo script de detección detectar_spoofingv1.sh se lanza el módulo de contención contencion_arp.py.

**Contencion_arp.py:** Este módulo tiene la misma arquitectura del módulo contencion_agentes.py, la única diferencia es que el fichero /etc/tesis/macspoof donde se almacena el formato de la mac cisco cambia de ruta y que este método no tiene que validar si la red tiene gestión por CLI, ya que para el método de detección previo es totalmente obligatorio que el dispositivo sea gestionable por línea de comandos.

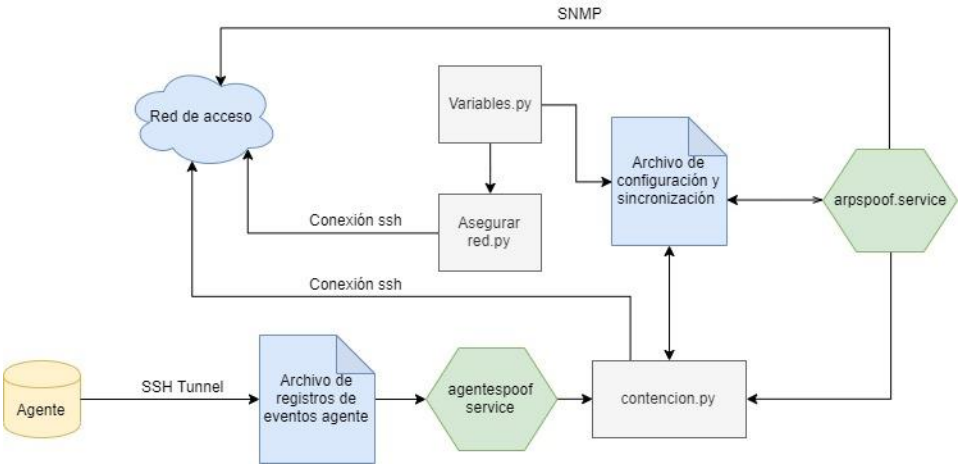


**Figura 65. Arquitectura resumen de detección y contención.**

Finalmente, sobre la figura 65 se ilustra la implementación de la arquitectura que permite dar cumplimiento al objetivo número 3 Implementar una arquitectura de atención de incidentes basados en ataques spoofing internos, la cual permita preparar la red, detectar los eventos de seguridad relacionados y tomar acciones automáticas de contención y mitigación.

## 2.3 Fase de validación.

En este capítulo se presenta el conjunto de pruebas que permitieron validar el correcto funcionamiento de la arquitectura de atención de incidentes basados en ataques spoofing internos diseñada e implementada en los capítulos anteriores, para ello se establece un escenario de laboratorio virtual en GNS3 v2.2.3 utilizando elementos similares a los presentados sobre el capítulo 1, en la figura 66 se ilustra el laboratorio implementado.

A continuación, se explicará cada uno de los elementos que compone la red de la figura 66.

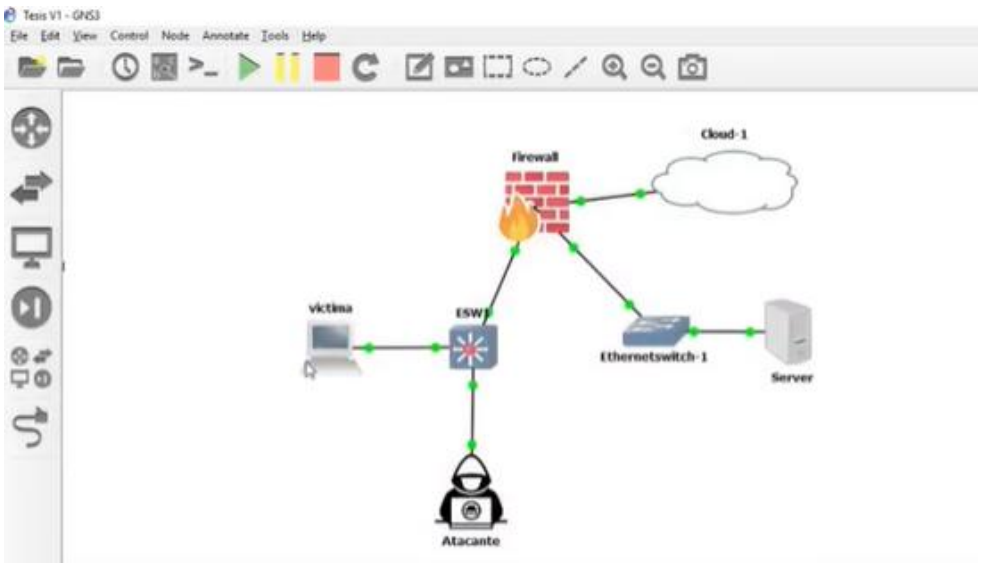


Figura 66. Laboratorio de pruebas.

- **Host Victima:** Host victima en sistema operativo Linux Ubuntu 18 sobre el cual se instalará el agente diseñado e implementado, sobre este equipo se realizarán los diferentes ataques tipo spoofing.
- **Host atacante:** Equipo atacante desde donde se materializarán los diferentes ataques, equipo con sistema operativo Kali Linux 2019.
- **ESW:** Switch cisco 3725 que gestiona el direccionamiento IP y actúa como switch de acceso para los anfitriones de la red, este es el dispositivo de red que debe ser asegurado sobre la fase de preparación implementada.
- **Firewall:** Firewall Fortigate 6.2.2 que se encarga de zonificar la red aislando el servidor SCGI de la red de acceso, este elemento se encarga de dar salida a internet y dentro de la topología no interviene en la protección de los ataques spoofing, ya que estos son perpetrados desde el interior de manera horizontal por lo que el firewall perimetral no tendría visibilidad de los mismos.
- **Cloud:** Esta conexión simula el acceso a internet mediante una conexión NAT.
- **Ethernetswitch:** este dispositivo tiene como único propósito enlazar el servidor SCGI a la red, no tiene ninguna influencia sobre la topología (Switch GNS3).
- **Server:** Este dispositivo aloja el servicio de la arquitectura diseñada para el SCGI, desde allí se realizarán las labores de detección, contención y notificación.

**Direccionamiento IP configurado sobre la arquitectura:**

- **IP SCGI:** 10.0.0.2
- **LAN:** 10.10.10.0/24
- **Direccionamiento L3 Sw cisco ←→ Firewall:** 10.10.1.0/30
- **DNS de la red:** 10.10.1.1

Una vez definido el laboratorio se realizaron las pruebas en el siguiente orden:

1. Aseguramiento de la red en ESW
2. Instalación de agente en host victima
3. Ejecución de ataques Spoofing
    a. ARP Spoofing
    b. DNS Spoofing
    **c.** DHCP Spoofing

### 2.3.1 Validación aseguramiento de la red.

Antes de proteger la red se despliega la configuración actual de ESW con el fin de evidenciar los cambios sobre la configuración del Switch una vez sea ejecutado desde el SCGI el script ingresar_configuracion.sh y asegurar_red.py

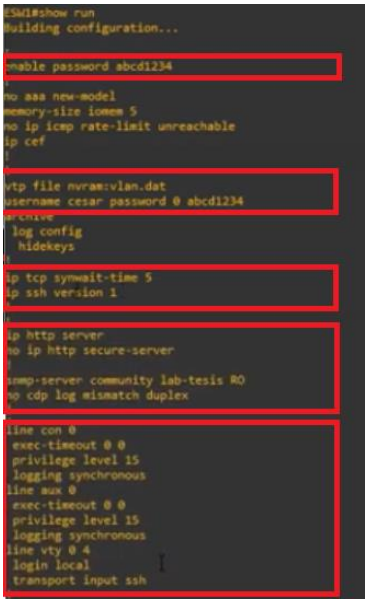


**Figura 67. Show running ESW desasegurado.**

En la figura 67 se observa rápidamente algunos parámetros desasegurados sobre la red de acceso:

1. Contraseñas de acceso al dispositivo en texto plano
2. Ssh en versión 1
3. Servicio http habilitado
4. Lina de acceso por consola sin protección de acceso ni contraseña
5. Acceso a las líneas de gestión remota SSH sin protección por lista de control de acceso vty

A continuación, se procede a ejecutar el script ingresar_configuracion.sh con la sentencia ./Ingresar_configuracion.sh y posterior respuesta a las diferentes preguntas que recaudan la información para el aseguramiento de la red.

```
root@ubuntu:/etc/tesis# ./Ingresar_configuracion.sh
¿Esta seguro que quiere iniciar el aseguramiento? Los archivos de configutacion seran reescritos s/n: s
¿Los dispositivos son administrable por CLI (SSH)? s/n: s
Cual es la direccion de red del segmento de administracion? (x.x.x.x/x): 10.0.0.0/24
Cual es el usuario administrador actual?: cesar
Cual es la contraseña de ingreso actual?: abcd1234
cual es o sera la vlan de gestion?: 1
Por buenas practicas no se debe utilizar la VLAN 1, por favor ingresar otra vlan
cual es o sera la vlan de gestion?: 10
Cual debe ser la red de gestion? (x.x.x.x/x): 10.10.1.0/30
La red tiene un diseño jerarquico? (s/n): n
Cual es la ip del Switch que gestiona las VLAN o el direccionamiento IP? (x.x.x.x): 10.10.1.2
La autenticacion sera por medio de AAA? (s/n): n
cual sera el usuario administrador?: soporte
cual sera la contrasela de administracion?: abcd1234
la clave no cumple con las requisitos minimos de seguridad
1. Debe contener minimo un caracter especial
2. Debe contener minimo un caracter en mayuscula
3. Debe contener minimo un caracter numerico
4. Debe contener minimo 8 caracteres
cual sera la contrasela de administracion?: Ab123.
la clave no cumple con las requisitos minimos de seguridad
1. Debe contener minimo un caracter especial
2. Debe contener minimo un caracter en mayuscula
3. Debe contener minimo un caracter numerico
4. Debe contener minimo 8 caracteres
cual sera la contrasela de administracion?: Tesis.ScGi2019
La clave cumple con los parametros minimos de seguridad
Cual es la IP del servidor NTP?: 10.0.0.10
El servidor NTP tiene autenticacion? (s/n): n
Cual es la IP servidor syslog?: 10.0.0.10
Dispone de recursos para gestionar y mantener un esquema de port-security (s/n): n
Los dispositivos de red adminten 802.1x (s/n): n
el dispositivo es Cisco o Aruba Networks (cisco/aruba): cisco
cual es la clave del usuario privilegiado? (enable/sys): abcd1234
Cual es la comunidad snmp?: lab-tesis
Cual es la red permitida para indagar snmp (x.x.x.x/x)?:
```

**Figura 68. Creación archivo de configuración y sincronización de scripts**

Se debe validar si el archivo de configuración y sincronización de scripts antes de su cifrado queda bien generado, en la figura 69 se observa como el archivo queda bajo el formato establecido para la operación de los diferentes módulos de Python y scripts en bash.

```
#Este archivo guarda las variables para integrar los diferentes modulos


cli#s
Usuario-actual#cesar
Clave-actual#abcd1234
Vlan-gestion#10
wildcard-gestion#0.0.0.3
mascara-gestion#255.255.255.252
red-gestion#10.0.0.0
Modelo-jerarquico#n
sw-core#10.10.1.2
radius#n
conf-user#soporte
Conf-password#Abcd1234*
ntp#10.0.0.2
clave-ntp#n
rsyslog#10.0.0.2
portsecurity#n
autenticacion#n
fabricante#cisco
old#1.3.6.1.2.1.4.22.1.2
enable_pass#abcd1234
snmpcomunity#lab-tesis
mascara-snmp#0.0.0.255
red-snmp#10.0.0.0
smtp#smtp.gmail.com
buzon#scgi.tesis@gmail.com
notificacion#cesar.rios50545@gmail.com
```

**Figura 69. Archivo de configuración y sincronización de scripts.**

Una vez generado el archivo se ejecuta el script asegurar_red.sh con el fin de validar si sobre el archivo de configuración del switch de la red se aplican las buenas prácticas definidas sobre la fase de preparación.

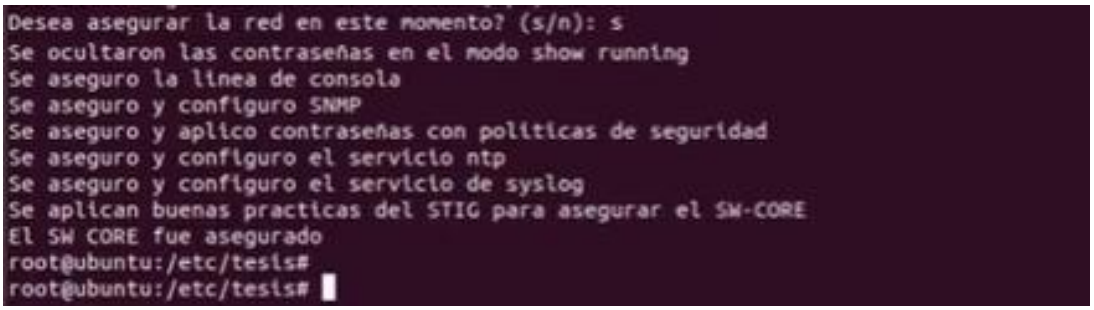


**Figura 70. Aplicando configuración sobre la red.**

En la figura 70 se observa como el script aplica las funciones establecidas para asegurar la red de manera automática y en la figura 71 se observa un fragmento de la configuración del switch previamente asegurado, apreciando las siguientes configuraciones:

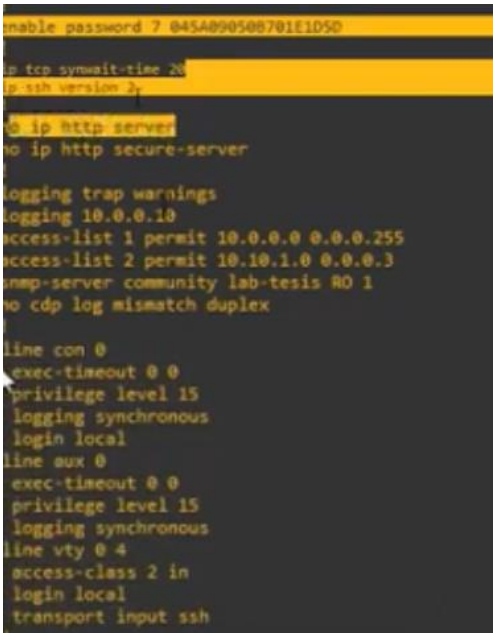


**Figura 71. Show running dispositivo asegurado.**

- Se ocultan las contraseñas del archivo de configuración.
- Se habilita ssh v2 y asegura el servicio
- Se deshabilita el servicio http
- Se habilita el loggin remoto

- Se crea la ACL para el aseguramiento de SNMP
- Se crea la ACL para el aseguramiento la las líneas VTY
- Se asegura el ingreso por consola

Para observar mejor el proceso se entrega el siguiente enlace https://youtu.be/cYtTWVpvIIU el comportamiento de la fase de preparación.

### 2.3.2 Validación instalación del agente en equipo protegido.

El script Instalar_agente.sh es el encargado de realizar la instalación del agente que protege la estación frente a ataques ARP, DNS y DHCP Spoofing, en la imagen 72 se observa la ejecución del script e instalación del agente.


Figura 72. Instalación del agente

Y finalmente la imagen 73 muestra el servicio detectarspoofing.service corriendo con normalidad en la estación protegida.

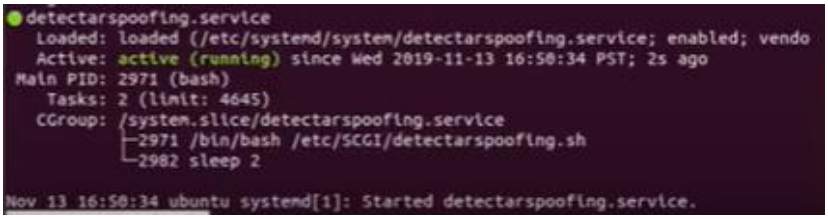


Figura 73. Detectarspoofing.service

En el siguiente enlace https://youtu.be/BAGptSm8HLM se puede observar el proceso de instalación del agente sobre la estación protegida.

### 2.3.3 Validación en ejecución de ataques spoofing

#### 2.3.3.1 Detección y contención de ARP Spoofing directamente desde el SCGI

Inicialmente se valida que el servicio de detección se encuentre corriendo sobre el SCGI mediante el comando “service arpspoof status”.

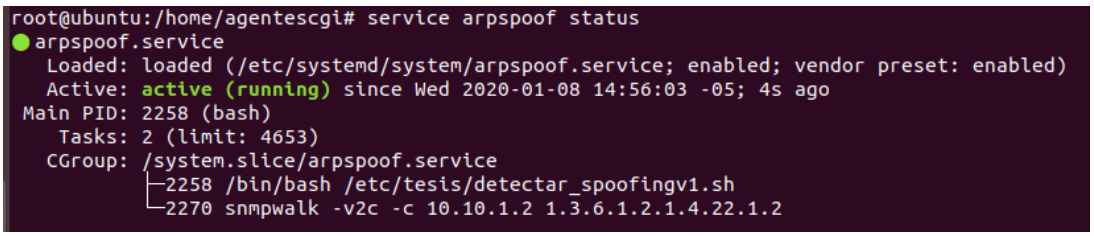


Figura 74. Arpspoof.service.

Adicional para esta prueba se utilizó la herramienta ettercap desde Kali Linux donde se realizaron varias pruebas teniendo el equipo atacante sobre interfaces diferentes en el switch de acceso, esto con el fin de validar el correcto funcionamiento de la fase de detección y contención.

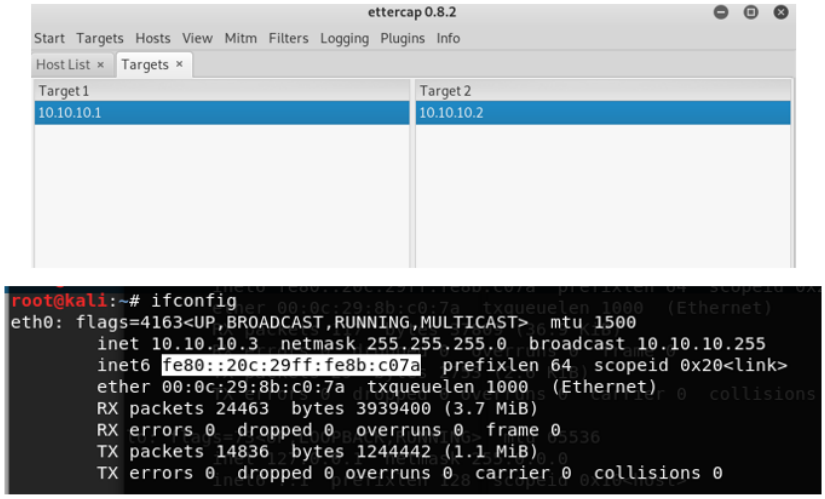


Figura 75. Configuración de ettercap.

- **Ataque realizado sobre la interface FastEthernet 1/2**


Figura 76. Contención interface FastEthernet 1/2

La figura 76 ilustra en la primera imagen, el evento en el switch para la acción realizada por el SCGI una vez detecta el ataque: La conexión desde la IP 10.0.0.2 y el cambio del estado de la interface FastEthernet1/2 a Down. La segunda imagen de la figura muestra la notificación enviada al buzón de corro del administrador para indicar las acciones realizadas.

- **Ataque realizado sobre la interface FastEthernet 1/5**

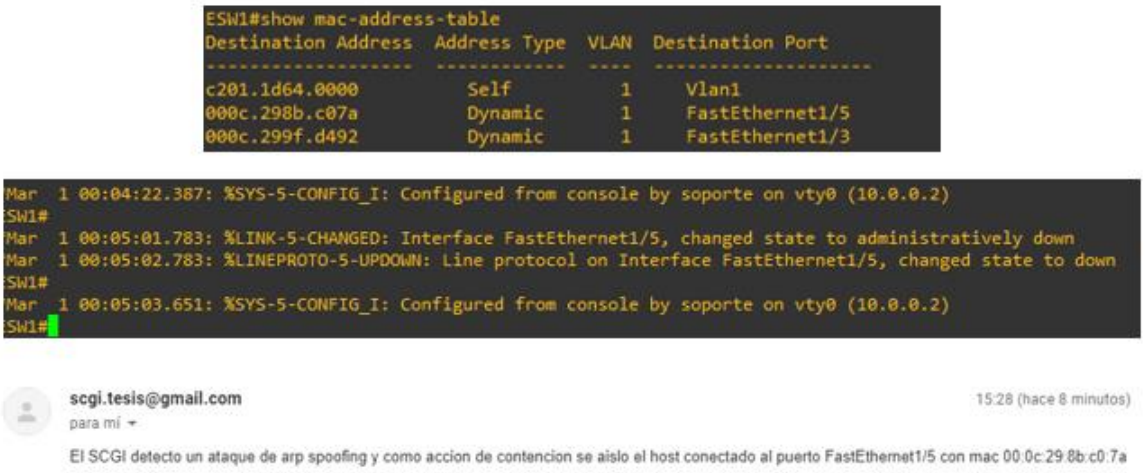


Figura 77. Contención interface FastEthernet 1/5

En las imágenes de la figura 77 se observa que la Mac de la interface de Kali Linux esta sobre la interface fastethernet 1/5 en el switch y que posterior a realizar el ataque se observa como el SCGI desde la ip 10.0.0.2 se conecta al switch y baja la interface 1/5 y posterior envía el correo de notificación.

- **Ataque realizado sobre la interface FastEthernet 1/8**


Figura 78. Contención interface FastEthernet 1/8.

- **Ataque realizado sobre la interface FastEthernet 1/11**


Figura 79. Contención interface FastEthernet 1/11

En las imágenes de la figura 79 se observa que la Mac de la interface de Kali Linux esta sobre la interface fastethernet 1/11 en el switch y que posterior a realizar el ataque se observa como el SCGI desde la ip 10.0.0.2 se conecta al switch y baja la interface 1/11 y posterior envía el correo de notificación.

Finalmente en la figura 80 se ve como el servicio arpsoof.service imprime en pantalla el registro del evento detectado y gestionado.


Figura 80. Log arpspoof.service.

El enlace https://youtu.be/CGMpb_gIRjA reproduce un video que ilustra el proceso de detección y contención en tiempo real.

### 2.3.3.2Detección y contención de ARP Spoofing desde agentes.

Inicialmente se valida que el servicio detectarspoofing este corriendo en la estación protegida y el servicio agestespoof en el SCGI.

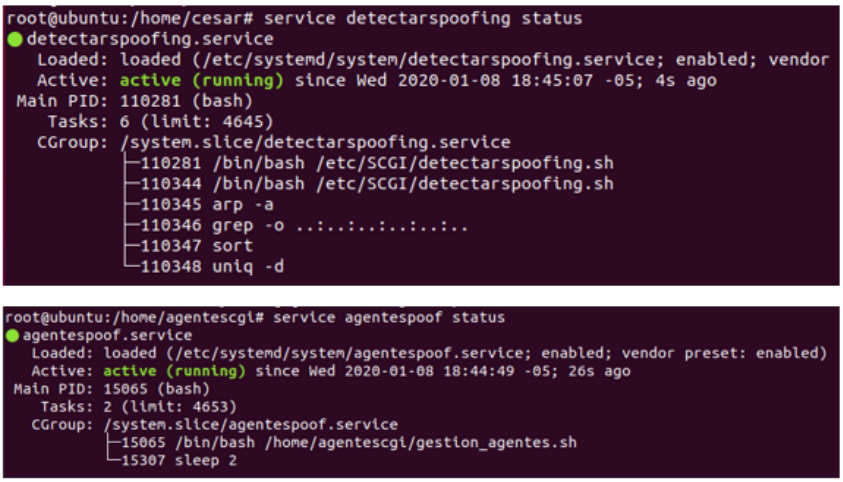


Figura 81. Servicios para gestión de agentes

La figura 81 muestra en la primera imagen el estado del servicio detectarspoofing corriendo en la estación protegida y en la segunda imagen el servicio agentespoof en el SCGI.

Después de validar los servicios se procede a realizar el ataque ARP Spoofing desde ettercap con Kali Linux, de igual manera, se realizará el ataque ubicando el equipo atacante en diferentes interfaces del switch con el fin de evidenciar el comportamiento de la arquitectura de atención de incidentes para ataques tipo spoofing.

- **Ataque realizado sobre la interface FastEthernet 1/4**

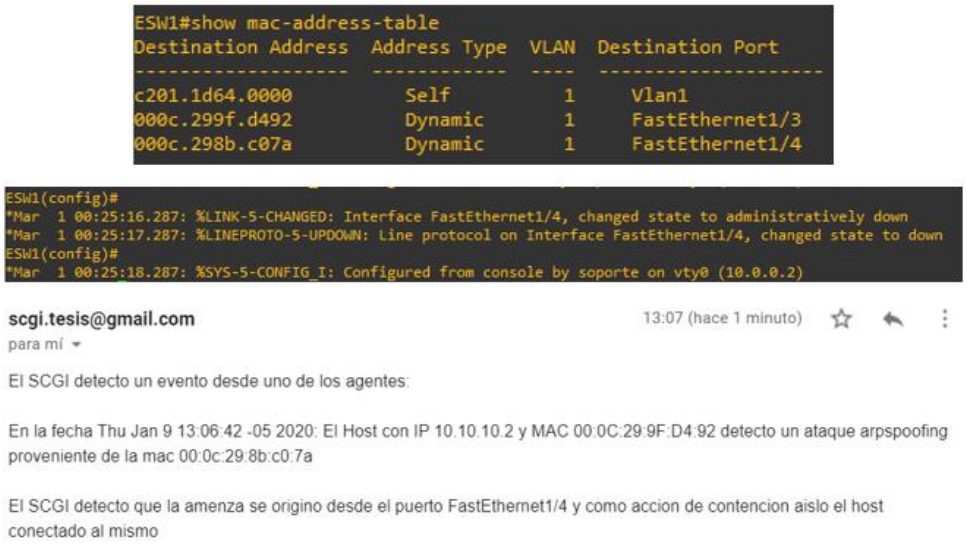


**Figura 82. Servicio agente: contención interface fastethernet 1/4**

La figura 82 muestra en la primera imagen la tabla MAC donde se observa la MAC del equipo atacante (c07a) sobre la interface fastethernet 1/4, la siguiente imagen muestra la acción realizada (apagado de puerto) desde el SCGI (10.0.0.2) una vez se detecta el ataque y finalmente la última imagen ilustra la notificación enviada por el SCGI al buzón de correo.

- **Ataque realizado sobre la interface FastEthernet 1/9**


**Figura 83. Servicio agente: contención interface fastethernet 1/9**

- **Ataque realizado sobre la interface FastEthernet 1/13.**


Figura 84. Servicio agente contención interface fastethernet 1/13.

Finalmente, cuando se gestiona un incidente, los eventos en el SCGI se ven de la siguiente manera.


Figura 85. Registro de eventos del servicio agentespoof.service

## 2.3.3.3Detección y contención de DNS Spoofing.

Para realizar las pruebas de DNS Spoofing se suplantará el DHCP y entregará los parámetros DNS falseados, indicándole al equipo victima que su servicio de resolución de nombres será a través de la IP del equipo atacante. En la figura 86 se expone la configuración IP del equipo atacante y en la figura 85 la configuración del módulo auxiliar de metasploit que corre el servicio DHCP Falseado.

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.10.4  netmask 255.255.255.0  broadcast 10.10.10.255
        inet6 fe80::20c:29ff:fe8b:c07a  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:8b:c0:7a  txqueuelen 1000  (Ethernet)
        RX packets 59285  bytes 14286975 (13.6 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 58719  bytes 10314606 (9.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 86. Configuración IP del equipo atacante

```
msf5 auxiliary(server/dhcp) > show options

Module options (auxiliary/server/dhcp):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   BROADCAST                    no        The broadcast address to send to
   DHCPIPEND                    no        The last IP to give out
   DHCPIPSTART                  no        The first IP to give out
   DNSSERVER   10.10.10.4       no        The DNS server IP address
   DOMAINNAME                   no        The optional domain name to assign
   FILENAME                     no        The optional filename of a tftp boot server
   HOSTNAME                     no        The optional hostname to assign
   HOSTSTART                    no        The optional host integer counter
   NETMASK     255.255.255.0    yes       The netmask of the local subnet
   ROUTER                       no        The router IP address
   SRVHOST     10.10.10.1       yes       The IP of the DHCP server
```

Figura 87. Configuración del servicio DHCP falso para suplantación de DNS.

Con el escenario preparado por parte del atacante se procede a correr el módulo auxiliar de metasploit con el comando run, seguidamente se valida los parámetros IP sobre la estación víctima y que son mostrados en la figura 88.

```
GENERAL.DEVICE:                         ens33
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         00:0C:29:9F:D4:92
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     Wired connection 1
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveCo
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         10.10.10.33/24
IP4.GATEWAY:                            10.10.10.1
IP4.ROUTE[1]:                           dst = 0.0.0.0/0, nh = 10.10.10.1, mt = 2
IP4.ROUTE[2]:                           dst = 169.254.0.0/16, nh = 0.0.0.0, mt =
IP4.ROUTE[3]:                           dst = 10.10.10.0/24, nh = 0.0.0.0, mt =
IP4.DNS[1]:                             10.10.10.4
IP6.ADDRESS[1]:                         fe80::9dc9:786a:5c26:9812/64
IP6.GATEWAY:                            --
IP6.ROUTE[1]:                           dst = fe80::/64, nh = ::, mt = 100
IP6.ROUTE[2]:                           dst = fe80::/64, nh = ::, mt = 256
IP6.ROUTE[3]:                           dst = ff00::/8, nh = ::, mt = 256, table
```

Figura 88. Configuración de los parámetros DNS en el equipo víctima.

El agente instalado en el equipo víctima detecta que el DNS no es el autorizado por el administrador de la red, envía el evento al SCGI estableciendo un túnel SCP por medio del protocolo SSH para que este gestione el incidente, la figura 89 ilustra el evento sobre los registros del servicio del agente.

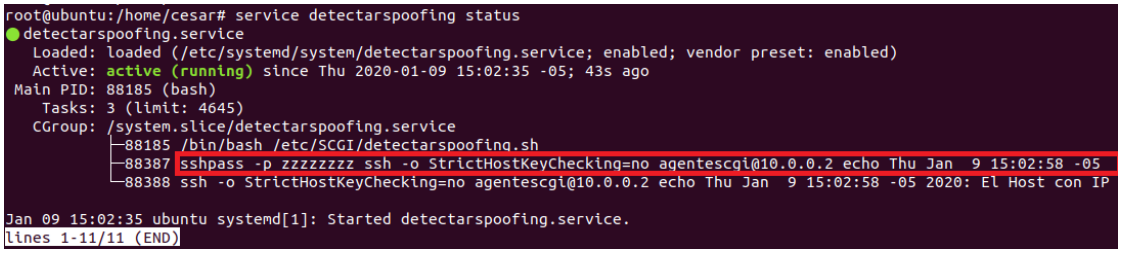


Figura 89. Envió de incidente desde equipo victima a SCGI.

Finalmente, el SCGI actúa ubicando la amenaza dentro de la red y aislando el equipo atacante de la misma.

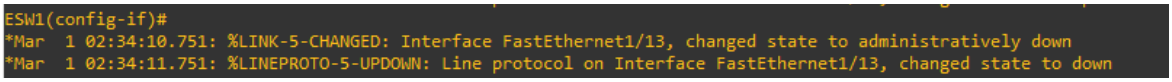


Figura 90. Aislamiento amenaza DNS.

**Ejecución de segunda prueba ubicando el atacante en la interface fastethernet 1/15**

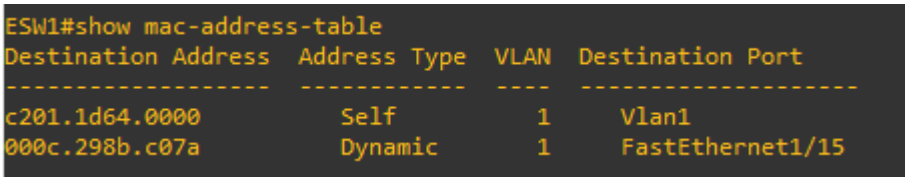


Figura 91. Atacante sobre interface fastethernet 1/15.

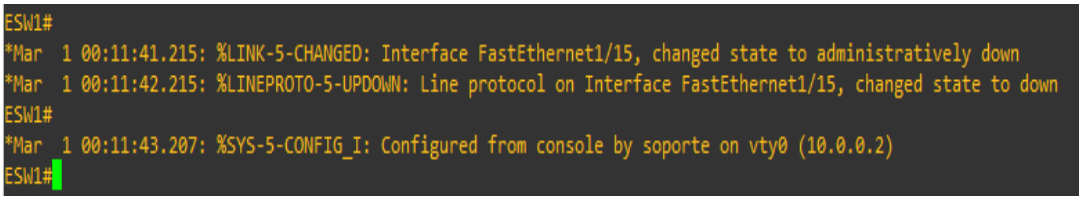


Figura 92. Contención de amenaza DNS en interface fastethernet 1/15

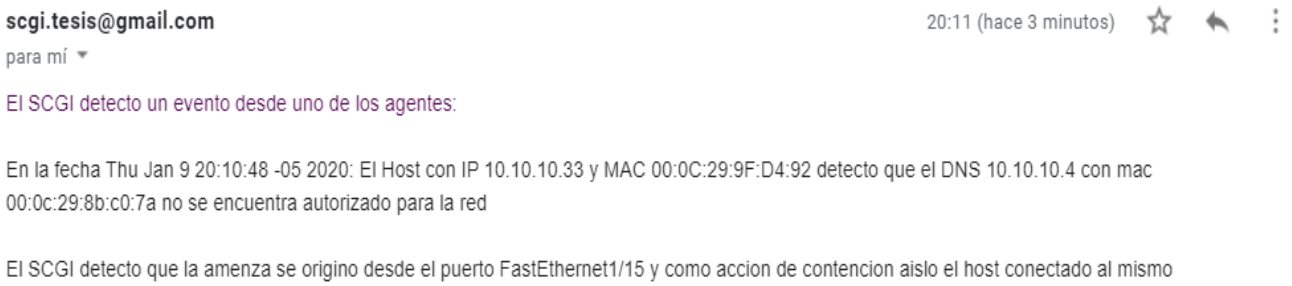


**Figura 93. Notificación de contención de amenaza DNS en interface fastethernet 1/15.**

El ataque es detectado y contenido según lo planeado y estructurado sobre la arquitectura, así mismo, sobre el enlace https://youtu.be/DNmVYdhpqRs se encuentra el video que permite ver este proceso en tiempo real.

## 2.3.3.4 Detección y contención de DHCP Spoofing.

Para este escenario el equipo atacante Kali Linux se aprovisiona con un servidor DHCP falso utilizando el módulo auxiliar de DHCP Server en metasploit, adicional, se habilita IP_forward para que se enruten las peticiones cuyo destino están en redes externas y un Source NAT en la tabla NAT del iptables con una cadena de POSTROUTING con el fin de habilitar la navegación a internet del equipo una vez este obtenga los parámetros IP entregados por el atacante, en la figura 94 se observa la arquitectura:

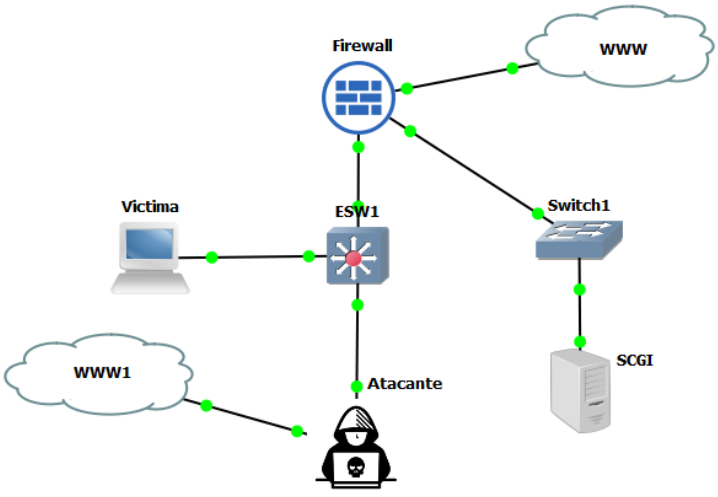


**Figura 94. Arquitectura para DHCP Spoofing**

Se observa en la figura 95 la configuración de las interfaces del equipo atacante, esta configuración tiene como propósito entregar un default Gateway falso a los anfitriones de la red con el fin de obligar a que el tráfico hacia otras redes sea dirigido al equipo atacante y en la figura 95 la configuración del enrutamiento y política de NAT.

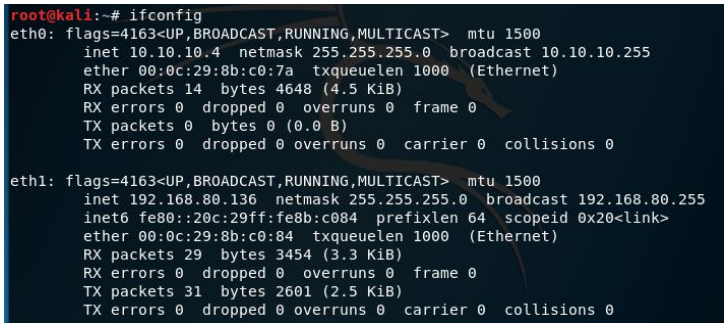


Figura 95. Configuración IP Atacante DHCP Spoofing


Figura 96. Configuración de Footing y NAT

Una vez preparado el equipo del atacante se procede a realizar el ataque mediante el uso del módulo auxiliar de metasploit auxiliary/server/dhcp.

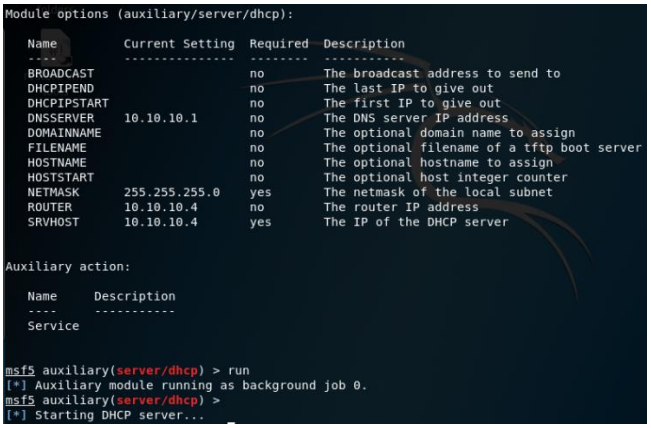


Figura 97. Configuración auxiliary/server/dhcp.

Segundos después de ejecutar el ataque, el equipo atacante pierde conectividad a nivel de capa de acceso L2 con la red.

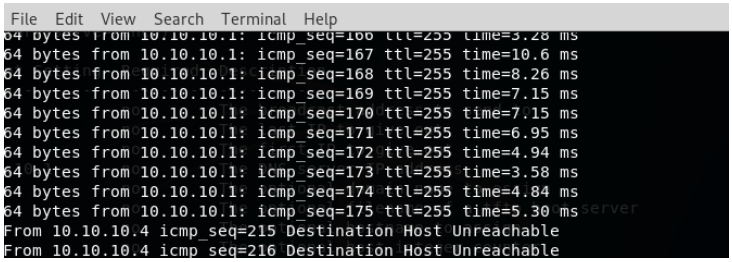


**Figura 98. Aislamiento del atacante**

El servicio del agente instalado en la estación protegida ilustra la detección y conexión con el SCGI a través de su IP publica (192.168.80.167).

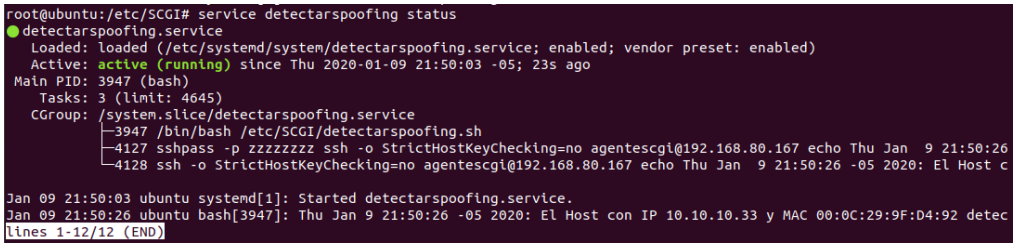


**Figura 99. Log DHCP Spoofing Agente**

Y al revisar el log en el Switch se idéntica que el puerto sobre el cual está el equipo atacante pasa a estado Down.

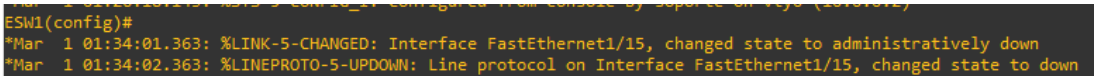


**Figura 100. Log DHCP Spoofing Switch**

Posteriormente llega el correo de notificación donde se informa al administrador las acciones ejecutadas desde el SCGI para detectar y aislar la amenaza.

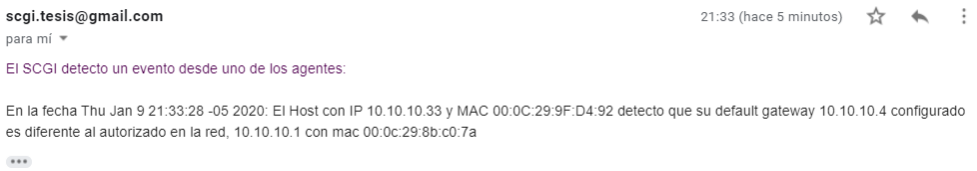


**Figura 101. Correo para acciones realizadas para DHCP Spoofing.**

**Ejecución de segunda prueba ubicando el atacante en la interface fastethernet 1/6**

```
ESW1#show mac-address-table
Destination Address  Address Type  VLAN  Destination Port
-------------------  ------------  ----  --------------------
c201.1d64.0000          Self          1    Vlan1
000c.299f.d492          Dynamic       1    FastEthernet1/3
0050.56c0.0005          Dynamic       1    FastEthernet1/3
0050.56c0.0003          Dynamic       1    FastEthernet1/6
000c.298b.c07a          Dynamic       1    FastEthernet1/6
```

Figura 102. MAC atacante FastEthernet1/6.

La imagen 102 evidencia que la MAC del equipo atacante Kali (c07a) esta sobre la interface FastEthernet1/6.

```
● agentespoof.service
   Loaded: loaded (/etc/systemd/system/agentespoof.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-01-09 21:49:27 -05; 1min 36s ago
 Main PID: 87843 (bash)
    Tasks: 3 (limit: 4653)
   CGroup: /system.slice/agentespoof.service
           ├─87843 /bin/bash /home/agentescgi/gestion_agentes.sh
           └─88580 python3.6 /root/eclipse-workspace/tesisV3/contencion_agentes.py

Jan 09 21:50:52 ubuntu bash[87843]: b'configure terminal\r\nEnter configuration commands, one per line.  End with CNTL/Z
Jan 09 21:50:52 ubuntu bash[87843]: FastEthernet1/6
Jan 09 21:50:52 ubuntu bash[87843]: b'interface FastEthernet1/6\r\nESW1(config-if)#'
Jan 09 21:50:52 ubuntu bash[87843]: b'clear arp\r\nESW1#'
Jan 09 21:50:52 ubuntu bash[87843]: El SCGI detecto un evento desde uno de los agentes:
Jan 09 21:50:52 ubuntu bash[87843]: En la fecha Thu Jan 9 21:50:26 -05 2020: El Host con IP 10.10.10.33 y MAC 00:0C:29:9
Jan 09 21:50:52 ubuntu bash[87843]: El SCGI detecto que la amenza se origino desde el puerto FastEthernet1/6 y como acci
Jan 09 21:50:54 ubuntu bash[87843]: el evento es Thu Jan 9 21:50:47 -05 2020: El Host con IP 10.10.10.33 y MAC 00:0C:29:
Jan 09 21:50:54 ubuntu bash[87843]: el numero del evento es 301
Jan 09 21:50:54 ubuntu bash[87843]: En la fecha Thu Jan 9 21:50:47 -05 2020: El Host con IP 10.10.10.33 y MAC 00:0C:29:9
lines 1-19/19 (END)
```

Figura 103. Evento agentespoof.service en SCGI

La figura 103 muestra el evento en el servidor SCGI que una vez detecta la notificación enviada desde la estación protegida acciona las labores de contención y notificación, aislando la amenaza conectada en la interface FastEthernet1/6 y posterior envió del correo.

**scgi.tesis@gmail.com** 21:50 (hace 0 minutos)

para mí

El SCGI detecto un evento desde uno de los agentes:

En la fecha Thu Jan 9 21:50:26 -05 2020: El Host con IP 10.10.10.33 y MAC 00:0C:29:9F:D4:92 detecto que su default gateway 10.10.10.4 configurado es diferente al autorizado en la red, 10.10.10.1 con mac 00:0c:29:8b:c0:7a

El SCGI detecto que la amenza se origino desde el puerto FastEthernet1/6 y como accion de contencion aislo el host conectado al mismo

Figura 104. Correo enviado DHCP Spoofing FastEthernet1/6.

# 3. Resultados

Se realizaron diferentes pruebas para los de ataques ARP Spoofing, DNS Spoofing y DHCP Spoofing, los cuales quedaron registrados de la siguiente manera:

- Cantidad de pruebas ARP Spoofing: 44
- Cantidad de pruebas DNS Spoofing: 248
- Cantidad de pruebas DHCP Spoofing: 20

En la figura 105 se observa la cantidad de eventos registrados por parte del SCGI para cada uno de los ataques realizados, básicamente, se están contando los eventos para cada módulo desarrollado sobre el archivo de notificaciones.

```
root@ubuntu:/home/agentescgi# cat notificaciones | grep -c arp
44
root@ubuntu:/home/agentescgi# cat notificaciones | grep -c DNS
248
root@ubuntu:/home/agentescgi# cat notificaciones | grep -c gateway
20
```

**Figura 105. Numero de pruebas realizadas.**

El porcentaje de eficacia para determinar el cumplimiento del objetivo general, crear una arquitectura que permita fortalecer y automatizar la gestión de incidentes de seguridad internos ante la preparación, detección y actuación de ataques ARP, DNS y DHCP spoofing, se definen con las siguientes formulas.

$$\%de\ aseguramiento = \frac{\#\ de\ cambios\ aplicados}{\#\ de\ cambios\ enviados} x100\%$$

$$\%de\ atencion\ de\ incidente = \frac{\Sigma ataques\ detectados}{\Sigma \text{ataques realizados}} x100\%$$

Una vez ejecutadas las diferentes pruebas se obtienen los siguientes resultados:

- Registro fase de aseguramiento.

$$\%de\ aseguramiento = \frac{18}{18} x100\%$$

$$\%de\ aseguramiento = 100\%$$

- Registro fase de deteccion y contencion.

$$\%de\ atencion\ de\ incidente = \frac{44 + 248 + 20}{44 + 248 + 20} x100\%$$

$$\%de\ atencion\ de\ incidente = 100\%$$

Se entrega en el Anexo 7 – notificaciones, el registro de los eventos de las pruebas realizadas, allí se observa el registro de los hallazgos identificados por los agentes.

Es así entonces, que con este conjunto de pruebas se logra dar cumplimiento al objetivo número 4. Generar un conjunto de pruebas el cual permita evaluar la arquitectura diseñada, tabulando los diferentes resultados esperados contra los objetivos propuestos, y dando cumplimiento al plan de trabajo establecido en la metodología de investigación.

# 4. Conclusiones y recomendaciones.

## 4.1 Conclusiones

Los ataques spoofing aprovechan las debilidades presentes en los modelos de comunicación diseñados décadas atrás, por tanto, existen diversos mecanismos de detección y contención en el mercado que apoyan la gestión de estos riesgos, sin embargo, muchas de estas soluciones hacen parte de la propiedad intelectual de varios fabricantes o exponen una dificultad alta en su gestión, desde cambios en la arquitectura de las tarjetas físicas de red, adopción de IPv6, gestión individual de las tablas ARP, configuraciones especiales en los dispositivos de red, entre otros. Adicional se encuentra la gestión asociada a la atención del incidente que permite controlar el impacto de la amenaza, exigiendo esfuerzos de personas y recursos.

Gracias a este trabajo de investigación e implementación de una arquitectura que fortalece y automatiza la gestión de incidentes de seguridad internos ante la preparación, detección y actuación de ataques ARP, DNS y DHCP spoofing es posible minimizar considerablemente el riesgo generado por este tipo de ataques, ya que el resultado de las pruebas realizadas fue bastante satisfactorio teniendo un porcentaje de atención y detección equivalente al 100%.

Adicional al entregar sobre una notificación la gestión realizadas y la identidad del equipo atacante es posible para el administrador de la red tomar acciones disciplinarías y legales

que permitan cerrar las brechas asociadas a la integridad de los empleados o personas internas de la organización.

Por otro lado, el desarrollo del módulo de aseguramiento automático de la red mediante la aplicación de buenas prácticas de STIG dejan una interesante manera de aplicar controles de cumplimiento, ya que su arquitectura permite traducir un requerimiento de negocio en una implementación técnica, por ello, esta metodología podría constituir una base importante para el desarrollo de futuras tesis en donde sea necesario aplicar controles de cumplimiento sobre una infraestructura tecnológica.

Adicional mediante el módulo de detección de ARP Spoofing desarrollado y entregado directamente sobre el servidor central SCGI se genera un aporte bastante importante ya que para ejecutar las diferentes acciones de detección y contención no es necesario instalar agentes, configuraciones especiales sobre los anfitriones de la red o poner en escucha un puerto spam sobre el switch, a diferencia de otro tipo de soluciones que requieren de este tipo de aprovisionamientos.

Finalmente, las acciones de contención desarrolladas en Python automatizan la respuesta del incidente permitiendo minimizar significativamente el impacto de la amenaza tipo Spoofing, esta clase de acciones hoy día generan mucho valor a las organizaciones tanto es, que los SOC (Security Operation Center) más experimentados del mercado iniciaron con la adopción de tecnologías SOAR (Security Orchestration, Automation and Response) que básicamente mediante Playbooks, scripts en Python, inteligencia de amenazas y herramientas de Big Data automatizan la respuesta a incidentes de una organización, sin embargo, SOAR al ser una tecnología en nacimiento tiene costos de adopción extremadamente altos los cuales son difícilmente adquiridos por una organización.

## 4.2 Recomendaciones.

- El desarrollo de la metodología de investigación entrega como producto final una arquitectura para automatizar la respuesta a incidentes de seguridad de la información relacionados con ataques internos mediante la ejecución de técnicas

spoofing, que después de las pruebas realizadas arrojo un porcentaje de eficiencia del 100%, sin embargo, dicha arquitectura no ha sido implementada sobre un ambiente productivo real en donde se pueda apreciar el comportamiento de la misma, por ello se recomienda implementar la solución sobre este ambiente.

- Dentro del alcance del desarrollo de la arquitectura se tuvo en cuenta sistemas operativos Linux, se recomienda implementar esta lógica, algoritmos y módulos sobre sistemas operativos Windows y aprovechar que el método de detección fue desarrollado desde scripts en bash para migrar el código a scripts en Power Shell y ejecutables .bat.
- El alcance de la arquitectura solo contemplo la creación de funciones con las sentencias del código para dispositivos cisco, sin embargo, la tabla de requerimientos para el aseguramiento de la red es trasversal a toda infraestructura de networking, por lo que se recomienda crear las funciones para otro tipo de fabricantes y ampliar el alcance de este trabajo.

# Anexos.

- Anexo 1: Funciones Tesis.py
- Anexo 2.1: Variables.py
- Anexo 2.2: Ingresar Configuracion.sh
- Anexo 3.1: Asegurar red.py
- Anexo 3.2: Asegurar red.sh
- Anexo 4: Instalar Agente.sh
- Anexo 5: Detectar SpoofingV1.sh
- Anexo 6: Contencion de agentes.sh
- Anexo 7: Log de Notificaciones

# Bibliografía

[1] Panda Security. (2020). *Las amenazas insider aumentan un 47%*. https://www.pandasecurity.com/spain/mediacenter/seguridad/cost-insider-threat-report/

[2] Hernandez, M. (2019). *Presentación Account Manager - PAM Senhasegura*.

[3] Institute, Ponemon. (2020). *2020 Cost of Insider Threats*. https://www.observeit.com/2020costofinsiderthreat/

[4] Ramba, J. L. G. (2017). *Ataques en redes de datos IPv4 e IPv6*. (0xWORD, Ed.) (3° Edición). España. ISBN: 978-84-617-9278-8

[5] Garrell, A., & Lloren Guilera. (2019). *La industria 4.0 en la sociedad digital* (M. Books ed; 1st). Barcelona España. ISBN 9788417313869

[6] Murray, G., Johnstone, M. N., & Valli, C. (2017). The convergence of it and ot in critical infrastructure. *Proceedings of the 15th Australian Information Security Management Conference, AISM 2017*, 149–155. https://doi.org/10.4225/75/5a84f7b595b4e

[7] Nozomi Networks. (2020). *Industrial Strength OT and IoT Security and Visibility*. https://www.nozominetworks.com/downloads/US/Nozomi-Networks-Guardian-Data-Sheet.pdf

[8] MINTIC. (2016). Guía para la Gestión y Clasificación de Incidentes de Seguridad de la, (21). Retrieved from https://www.mintic.gov.co/gestionti/615/articles-5482_G21_Gestion_Incidentes.pdf

[9] cisco-edges-competition-by-being-a-gartner-wired-and-wireless-lan-access-magic-quadrant-leader @ blogs.cisco.com. (n.d.). Retrieved from

https://blogs.cisco.com/wireless/cisco-edges-competition-by-being-a-gartner-wired-and-wireless-lan-access-magic-quadrant-leader

[10] Cisco. (2019). Hardering netowrok cisco. Retrieved from http://www.cisco.com/c/en/us/support/docs/ip/access-lists/13608-21.html

[11] Networks, A. (2019). *ARUBA OS H ARDENING GUIDE*. https://support.arubanetworks.com/Documentation/tabid/77/DMXModule/512/EntryId/16036/Default.aspx

[12] Stig. (2019). stigs @ www.stigviewer.com. Retrieved from https://www.stigviewer.com/stigs

[13] NIST. (2014). Security and Privacy Controls for Federal Information Systems and Organizations Security and Privacy Controls for Federal Information Systems and Organizations. *Sp-800-53Ar4*, 400+. https://doi.org/10.6028/NIST.SP.800-53Ar4

[14] Rupal, D. R., Satasiya, D., Kumar, H., & Agrawal, A. (2016). Detection and prevention of ARP poisoning in dynamic IP configuration. *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. https://doi.org/10.1109/RTEICT.2016.7808030

[15] Pal Singh, A., & Deep Singh, M. (2014). Analysis of Host-Based and Network-Based Intrusion Detection System. *International Journal of Computer Network and Information Security*, *6*(8), 41–47. https://doi.org/10.5815/ijcnis.2014.08.06

[16] Response, A., & Capabilities, F. (2017). Using splunk adaptive response. Retrieved from https://www.splunk.com/pdfs/technical-briefs/using-splunk-adaptive-response.pdf

[17] Miller, D. (2015). Security Information and Event Management. Retrieved from https://www.mcafee.com/mx/resources/solution-briefs/sb-siem-integrated-endpoint-security.pdf

[18] DarkTrace (2020) Cyber AI Platform Anrigena from https://www.darktrace.com/en/technology/

[19] Plummer, D. C. (1982). rfc826 @ tools.ietf.org. Retrieved from https://tools.ietf.org/html/rfc826

[20] Ruiz, M. Á. H. (2013). *ARP Spoofing*. https://www.incibe-cert.es/blog/arp-spoofing

[21] Ciyi, C. G. (2010). Hablemos de Spoofing. Retrieved from http://hacking-etico.com/2010/08/26/hablemos-de-spoofing/

[22] Patel, R. S. (2013). *Kali Linux social Engineering*. (Livery Place, Ed.). Birmingham: Pack Publishing Ltd.

[23] David W. Cooke, R. (2014). *No TitleThe Resilience of the Electric Power Delivery System in Response to Terrorism and Natural Disasters*. Washington, D.C. Retrieved from https://www.nap.edu/read/18535/chapter/4#11

[24] Arslan, Y. (2017). *A solution for ARP spoofing: Layer-2 MAC and protocol filtering and arpserver*, from https://arxiv.org/ftp/arxiv/papers/1708/1708.01302.pdf

[25] Merino, B. (2012). Defensas frente a ataques DHCP. Retrieved March 7, 2012, from https://www.securityartwork.es/2012/03/07/defensas-frente-a-ataques-dhcp/

[26] H. M. Elshafie, T. M. Mahmoud and A. A. Ali, (2019) "Improving the Performance of the Snort Intrusion Detection Using Clonal Selection," International Conference on Innovative Trends in Computer Engineering (ITCE).

[27] Nikhil Tripathi, BM Mehtre. (2014). International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT). Analysis of Various ARP Poisoning Mitigation Techniques: A comparison, School of Computer and Information Sciences, University of Hyderabad, Hyderabad, India.

[28] Cordero, M., Viñamagua, M., & Garzón, C. (2015). *Detección y mitigación de ataques ARP Spoof empleando entornos virtualizados*. *6*, 31–37. https://journal.espe.edu.ec/ojs/index.php/geeks/article/download/282/261

[29] June, M., No, I., & Bijral, R. K. (2017). Study of Vulnerabilities of ARP Spoofing and its detection using SNORT. *International Journal of Advanced Research in Computer Science*, *8*(5), 2074–2077. http://www.ijarcs.info/index.php/Ijarcs/article/view/4016/3667

[30] Rao, G. R. K., & Prasad, R. S. (2018). Shielding the networks depending on linux servers against ARP spoofing. *International Journal of Engineering and Technology(UAE)*, *7*(2), 75–79. https://doi.org/10.14419/ijet.v7i2.32.13531

[31] Thelen, D. G., Martin, J. A., & Roth, J. D. (2019). *Patent Application Publication: US 2019 / 0125831 A1*. https://patentimages.storage.googleapis.com/e8/80/ff/4c58c40b0496d7/US20190058731A1.pdf

[32] Coleman, A., Bombal, D., & Duponchelle, J. (2019). *Getting Started with GNS3*. https://docs.gns3.com/1PvtRW5eAb8RJZ11maEYD9_aLY8kkdhgaMB0wPCz8a38/index.html#h.36ilounhvvlx

[33] Bombal, D., Grossmann, J., Duponchelle, J., & Blackwell, M. (2018). *What's new in GNS3 version 2.0*.

https://docs.gns3.com/1jtdTQAcKa7JmQTNH2LoxQmOYalts7O0urmZ9CNnoEpU/index.html#h.trg3qh53krs9

[34] *QEMU documentation*. (2020). https://wiki.qemu.org/Main_Page

[35] GnuPG, G. (2019). *GnuPG 2.2.19 released*. https://lists.gnupg.org/pipermail/gnupg-announce/2019q4/000443.html

[36] ] J. Case, M. Fedor, M. Schoffstall, J. davi. (1990). rfc1157 @ www.ietf.org. Retrieved from https://www.ietf.org/rfc/rfc1157.txt

[37] Mockapetris, P. (1987). Rfc1034 @ Www.Ietf.Org. Retrieved from https://www.ietf.org/rfc/rfc1034.txt

[38] Foundation, F. S. (2017). *GNU Bash*. http://www.gnu.org/software/bash/bash.html

[39] Marzal Varó, A., Gracia Luengo, I., & García Sevilla, P. (2014). Introducción a la programación con Python 3. In *Introducción a la programación con Python 3*. https://doi.org/10.6035/sapientia93

[40] Coleman, A., Bombal, D., & Duponchelle, J. (2019). *Which emulator should I use?* https://docs.gns3.com/1o4IX8nXISl5gb4BwoSFrUht3MeTjkzHM1TCeWAe669g/index.html#h.dkuh6r6yqch5

[41] Python. (2020). *The Python Standard Library*. The Python Software Foundation is a non-profit corporation. Retrieved from https://docs.python.org/3/library/

[42] Paramiko (2020). *Welcome to Paramiko A Python implementation of SSHv2.* Powered by Sphinx 1.6.7 & Alabaster 0.7.12. Retrieved from http://www.paramiko.org/

[43] Hunt J. (2019) *Functions in Python*. In: A Beginners Guide to Python 3 Programming. Undergraduate Topics in Computer Science. Springer, Cham

[44] Carlos Álvarez Martín y Pablo González Pérez. (2015). *Hardening de servidores GNU / Linux.3ra edición* SBN: 978-84-617-1518-3 (0xWORD, Ed.). España.

[45] Unix (2014). *man nmcli nmcli(1) [centos man page]*. Retrieved from https://www.unix.com/man-page/centos/1/nmcli/