Implementación y pruebas de monitoreo en una red LAN, basados en SNMPv3.

Ochoa E.; Venegas M.; Medina W. Mgs.
Facultad de Ingeniera en Electricidad y Computación
Escuela Superior Politécnica del Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 vía Perimetral
Apartado 09-01-5863. Guayaquil-Ecuador
eeochoa@espol.edu.ec, mapavene@espol.edu.ec, wmedina@espol.edu.ec

Resumen

El objetivo de este proyecto es acercar a cualquier usuario a los beneficios de SNMPv3, cuya nueva estrategia de seguridad es la incorporación de autenticación y cifrado de los mensajes que son generados por las aplicaciones de monitoreo. Además se pretende facilitar la comprensión de la instalación, configuración de los agentes SNMPv3 y mediante ejemplos sencillos se demuestra cómo el protocolo genera las llaves con las cuales los mensajes son asegurados y cómo junto a los usuarios éstas son almacenadas al interior del agente de forma segura. Esta versión es muy sencilla de implementar en cualquiera de los dispositivos que forman parte de las redes hoy en día y sin embargo, no está implementada en la gran mayoría de los entornos informáticos, precisamente por el desconocimiento de su despliegue y operación.

Palabras Claves: *SNMPv3, aplicación, agente, llaves, autenticación, privacidad.*

Abstract

The main objective of this project is give an approach to any user to the benefits of SNMPv3, which has a new strategy of security that is the incorporation of authentication and encryption of the messages generated by the monitoring applications. This project also eases the comprehension of how to install, configure SNMPv3 agents and also using easy examples it demonstrates how the protocol generates the keys that secure the messages and how together with the users configured, keys are stored in a very secure way within the agent. Despite the implementation of this version is very simple in any current network device, it is not implemented in most of the cases, because the lack of knowledge of its deployment and operation.

Keywords: *SNMPv3, application, agent, keys, authentication, encryption.*

1. Introducción

El protocolo SNMP es el núcleo de las operaciones de administración generadas por cualquiera de las aplicaciones de uso libre o con licenciamiento, que son instaladas en las redes corporativas de cualquier tipo para su monitoreo. En su gran mayoría son las primeras dos versiones de SNMP las que son configuradas en los equipos de organizaciones, incluso en aquellas que por su ocupación deben tener un alto grado de privacidad en sus mensajes. Sin embargo, aquellas versiones utilizan el esquema de comunidades el cual es vulnerable. La última versión del protocolo mejora considerablemente la seguridad de los mensajes al someterlos a un proceso de autenticación y cifrado de su contenido al ser enviados y recibidos por los agentes dentro de la red.

En la actualidad el conocimiento en las empresas sobre la instalación y manejo de SNMPv3 es muy escaso, más que nada, de las características nuevas de seguridad y cómo operan en la última versión del protocolo, que paradójicamente fue introducido como estándar con el esquema de seguridad USM (Modelo de Seguridad basado en Usuarios) desde el año 2002. Durante el desarrollo del presente documento, se irá presentando de forma clara y sencilla los principales aspectos del funcionamiento del protocolo, por ejemplo la manera en que un mensaje saliente es autenticado y sometido a cifrado y los parámetros que lo acompañan, para que el agente destino pueda autenticarlo y descifrar su contenido. De esta forma cualquier usuario podrá implementar SNMPv3 en su red de forma sencilla y sin ningún tipo de gasto adicional.

2. Conceptos importantes.

2.1. Arquitectura SNMPv3

Un agente SNMPv3 está compuesta por el motor SNMP y las aplicaciones.

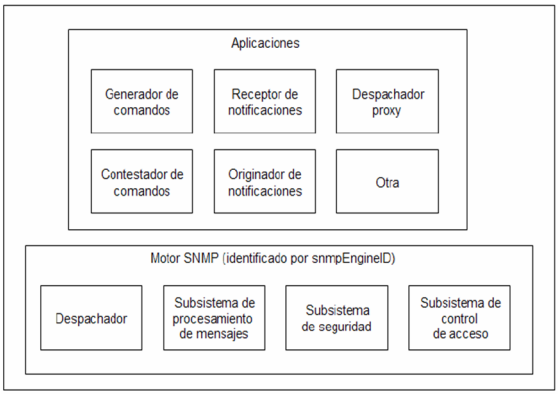

Figura 1. Diagrama de un agente SNMPv3. [7]

El motor está compuesto por 4 módulos: despachador, subsistema de procesamiento de mensajes, subsistema de seguridad y subsistema de control de acceso.

El despachador verifica la versión del mensaje SNMP recibido antes de enviarlo al subsistema de procesamiento de mensajes; y además envía mensajes hacia motores externos. El subsistema de procesamiento contiene algunos sub módulos, por ejemplo, para procesar solicitudes de mensajes exclusivas para cada una de sus tres versiones; extrae la información de los mensajes recibidos y termina de preparar los que serán enviados. El subsistema de seguridad provee servicios de autenticación basados en comunidades o en usuarios, la de usuarios hace uso de los algoritmos MD5 (Algoritmo de resumen de mensaje v5) o SHA (Algoritmo de hash seguro) para autenticar los mensajes sin transmisión de contraseñas en texto plano. Este subsistema también provee servicios de privacidad para cifrar el contenido de los mensajes SNMP enviados haciendo uso de algoritmos como DES (Estándar de Cifrado de Datos) o AES (Estándar Avanzado de Cifrado). Finalmente, el subsistema de control de acceso permite definir a qué objetos un usuario puede acceder dentro de la MIB y qué operaciones puede realizar en ellos.

Las aplicaciones incluyen:

- Generador de comandos: Implementada por un equipo administrador de red, genera solicitudes de lectura y escritura para los agentes monitoreados y procesa sus respuestas.
- Respondedor de comandos: Implementada por un dispositivo monitoreado, responde a las diferentes solicitudes que recibe desde un equipo administrador de red.
- Originador de notificaciones: Genera traps para ser enviadas al equipo administrador de red.
- Receptor de notificaciones: Reside en el equipo administrador de red, procesa las traps recibidas.
- Enviador proxy: Facilita el paso de mensajes entre agentes. [1]

2.2. Modelo de Seguridad basado en usuarios (USM).

Utiliza el concepto del motor SNMP autoritativo (AuthoritativeEngine), el que sirve como referencia para el control de la sincronización entre agentes monitoreado y administrador.

En todo proceso de transmisión de mensajes, uno de aquellos dos agentes, tendrá designado el motor SNMP autoritativo, dependiendo de dos reglas: si el agente realiza una solicitud esperando una respuesta, el motor SNMP del agente receptor será autoritativo; si en cambio un agente recibe un mensaje sin solicitarlo (trap, response, report), el motor SNMP del agente transmisor será autoritativo.

El modelo USM tiene tres módulos con diferentes especificaciones:

- Módulo de autenticación: Este módulo se asegura de que el mensaje provenga de la fuente de donde dice ser generado y que en el proceso del envío no haya sido modificado. Para la comunicación entre agentes es necesario compartir una llave de autenticación.
- Módulo timeliness: Es el que proporciona la protección contra el retraso o retransmisión maliciosos de los mensajes. El agente administrador copia unos parámetros del agente para la sincronización (snmpEngineBoots y snmpEngineTime). Por medio de estos valores el agente administrador puede hacer una estimación del tiempo en que los mensajes van a llegar al agente, con la cual se decide si se descarta o no un mensaje dependiendo si existe una diferencia en más o en menos de 150 segundos, esta estimación de tiempo se guarda en una memoria no volátil del agente.
- Módulo de privacidad: Es el que proporciona el cifrado y descifrado del contenido (datos) de un mensaje. Para realizar el cifrado del mensaje se necesitan los datos, una llave de privacidad y un vector de inicialización (en adelante, IV, por sus siglas en inglés) cuya formación se explica más adelante.

Según lo establecido por el protocolo SNMP no es posible realizar el cifrado de los datos sin realizar la autenticación y verificación temporal primero. [4],[5],[6]

2.3. Algoritmos para la autenticación del mensaje.

Los protocolos usados por SNMPv3 para la autenticación del usuario están basados en algoritmos de hash y son los siguientes:

- MD5, Message Digest Algorithm v5.
- SHA, Secure Hash Algorithm.

Un algoritmo hash es una función computacional, llamada también de resumen, por medio de la cual, a una entrada determinada (por ejemplo una contraseña, un documento de texto, archivo, etc) de cualquier longitud, se producirá una salida de tamaño fijo, totalmente diferente para cada entrada. Otra característica de una función hash es que es imposible reconstruir el texto origen a partir del valor resumen o resultado, además de que idealmente no habrá colisiones, es decir que a valores diferentes de entradas no se puede tener un mismo hash de salida. Obviamente, en la vida real, los algoritmos de hash sí pueden generar colisiones debido a que el conjunto de valores de entrada a la función es infinito, es decir, puedes ingresar cualquier contraseña de cualquier extensión para obtener siempre una salida de tamaño fijo.

Por ejemplo, el MD5 genera una salida de 128 bits en la forma de 32 dígitos hexadecimales. Para que no haya colisiones, el número de posibles entradas debería estar limitado a 2^{128} palabras, pero como ya se mencionó las posibilidades de ingreso son infinitas. Lo que se trata con los algoritmos de hash modernos, es que sean lo suficientemente complejos, que las colisiones sean las mínimas posibles y muy difíciles de hallar. [10]

2.3.1. MD5 y SHA. El proceso de autenticación de un mensaje saliente comienza cuando dos llaves se originan de la llave secreta authKey de 16 octetos para MD5 y de 20 octetos para SHA. Se comienza por extender a 64 octetos la authKey al aumentarle 48 octetos de ceros en el caso de MD5 y 44 en SHA, al resultado se le llama extendedAuthKey; se origina la cadena de octetos de relleno interior IPAD (inner padding) al replicar el octeto 0x36 64 veces y a la operación XOR entre el extendedAuthKey e IPAD se le denomina llave K1.

Luego se origina la cadena de octetos de relleno exterior OPAD (outer padding) al replicar el octeto 0x5c 64 veces y se obtiene la llave K2 haciendo la operación XOR entre el extendedAuthKey y el OPAD.

Se antepone la llave K1 al mensaje y a todo esto se le calcula el resumen MD5 o SHA, dependiendo cuál se haya escogido previamente para autenticar los mensajes. Al resultado de la operación se le antepone la llave K2 y se vuelve a computar el resumen final MD5 de 128 bits (16 octetos) o SHA de 160 bits (20 octetos), del cual en ambos casos los primeros 96 bits (12 octetos) son el código de autenticación del mensaje (MAC) e irán en el campo msgAuthenticarionParameters del mensaje saliente.

Cuando un agente recibe un mensaje que requiere autenticación, extrae el valor que viene en el parámetro msgAuthenticarionParameters y lo reserva; hace el mismo procedimiento para hallar las llaves K1 y K2 y con ellas calcula el resumen MD5 o SHA como se describió. De la misma forma, los primeros 12 octetos del resumen, se colocan en el campo msgAuthenticarionParameters, además de compararlos con el valor que se tenía en reserva. Si ambos coinciden, el mensaje es autenticado correctamente y se procede entonces a verificar la validez temporal. [4]

2.4. Algoritmos de seguridad en el cifrado del mensaje

Los protocolos usados por SNMPv3 para el cifrado del campo *scopedPDU* del mensaje son DES o AES, el cual usa llaves de 128 bits.

El algoritmo DES, al poseer una clave relativamente corta no es muy fuerte para el cifrado por lo que ha sido reemplazado por AES ya que hasta ahora, se mantiene como un algoritmo muy seguro y que ha resistido a los ataques.

2.4.1. DES. Se vale para sus operaciones de la llave DES y el IV, que es una cadena de bytes usada de apoyo para las operaciones iniciales de cifrado y descifrado.

Los primeros 8 octetos de la llave secreta de cifrado privKey de 16 octetos se usan como la llave DES de la función hash para cifrar el mensaje; pero los bits menos significativos de cada octeto se descartan antes, ya que el algoritmo trabaja con llaves de 56 bits. Los 8 últimos octetos de la llave privKey, se usan como pre vector de inicialización (en adelante, pre-IV).

Una forma de asegurar que el IV no se repita para dos paquetes diferentes cifrados a partir de la misma llave, se necesita "condimentar" al pre-IV con algo único por paquete. Se llamará "condimento" a un valor de 8 octetos formado por la concatenación de 4 octetos correspondientes a snmpEngineBoots seguidos por 4 octetos que el motor local mantiene, generados de forma arbitraria al momento de booteo. Luego se hace la operación XOR entre el condimento y el pre-IV para formar el IV. El "condimento" se coloca en el campo privParameters del mensaje para habilitar al agente receptor calcular el IV correcto y así poder descifrar el mensaje. [4]

Cifrado del mensaje

El cifrado se realiza en modo Cipher Block Chaining - CBC por sus siglas en inglés (Encadenamiento de Bloques de Cifrado). Los datos a cifrar, específicamente el scopedPDU del mensaje (contextEngineID, contexName y datos), se dividen en cadenas múltiplos de 8 octetos (64 bits), si es necesario se hace un relleno al final para cumplirlo.

A cada bloque de texto plano (plaintext) se le hace un XOR con el texto cifrado (ciphertext) previo, el resultado se somete al algoritmo de cifrado y la salida de este proceso será el texto cifrado para el presente bloque. El procedimiento sigue hasta que ya no queden más bloques de texto plano. Para el primer bloque de texto plano, el IV es usado en vez de un texto cifrado de algún bloque previo, ya que no lo hay.

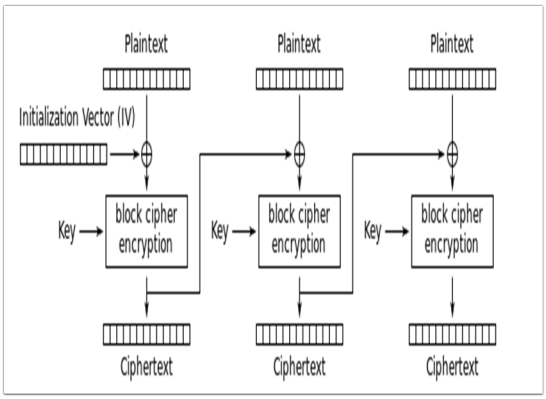

Figura 2. Cifrado DES en modo Encadenamiento de Bloques de Cifrado. [9]

Descifrado del mensaje.

El primer bloque de texto cifrado es sometido al algoritmo de descifrado, a la salida del descifrado se le hace un XOR con el IV y el resultado es el primer bloque de texto plano. Para cada bloque siguiente, el bloque de texto cifrado se descifra, a la salida del descifrado se le hace un XOR con el bloque de texto cifrado previo y el resultado es el bloque de texto plano.

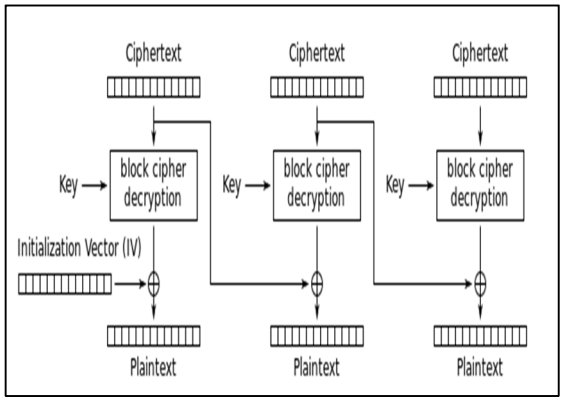

Figura 3. Descifrado DES en modo Encadenamiento de Bloques de Cifrado. [9]

2.4.2. AES. Se vale para sus operaciones de la llave de cifrado AES y el IV.

Los 128 bits (16 octetos) de la llave localizada se usan como la llave AES de la función hash para cifrar el mensaje. El vector de inicialización (IV) es la concatenación de los siguientes valores: el valor de 32 bits correspondiente a snmpEngineBoots del motor autoritativo, 32 bits correspondientes a snmpEngineTime y 64 bits correspondientes a un valor generado de forma pseudo-aleatoria al momento de booteo por el motor local. Estos 64 bits son el "condimento" de los mensajes y son colocados dentro del campo msgPrivacyParameters como cadena de octetos para permitir que el agente receptor descifre el mensaje (ya que al estar ambos agentes sincronizados, comparten los valores del módulo timeliness y solo les falta este valor para generar el IV). [5]

Cifrado del mensaje

El cifrado se realiza en modo Cipher Feedback - CFB (Cifrado por Retroalimentación). Los datos a cifrar, específicamente el *scopedPDU* del mensaje (contextEngineID, contexName y datos), se dividen en bloques de 128 bits (16 octetos), el último bloque podría tener menos de 128 bits pero no se requiere relleno.

Se aplica la operación de cifrado ($CIPH_K$) al IV para producir el primer bloque de salida (output block), al cual se le hace un XOR con el primer bloque de texto plano (plaintext) produciendo así el primer bloque de texto cifrado (ciphertext).

El bloque de texto cifrado se usa como bloque de entrada a la siguiente operación de cifrado. El último bloque de texto cifrado se obtiene haciendo un XOR entre el último bloque de texto plano de r bits (r puede ser menor a 128 bits) y los r bits más significativos del último bloque de salida.

Descifrado del mensaje.

El IV es el primer bloque de entrada, a la función de cifrado inversa ($CIPH^{-1}_K$). El primer bloque de texto cifrado se usa para el segundo bloque de entrada, el segundo texto cifrado se usa para el tercer bloque de entrada y así sucesivamente. La función de cifrado se aplica a cada bloque de entrada para producir los bloques de salida a los cuales se les aplica un XOR con los correspondientes bloques de texto cifrado para recuperar los bloques de texto plano. Al último bloque de texto cifrado de r bits (r puede ser menor a 128 bits) se le hace un XOR con el segmento de los r bits más significativos del último bloque de salida para obtener el último bloque de texto plano.

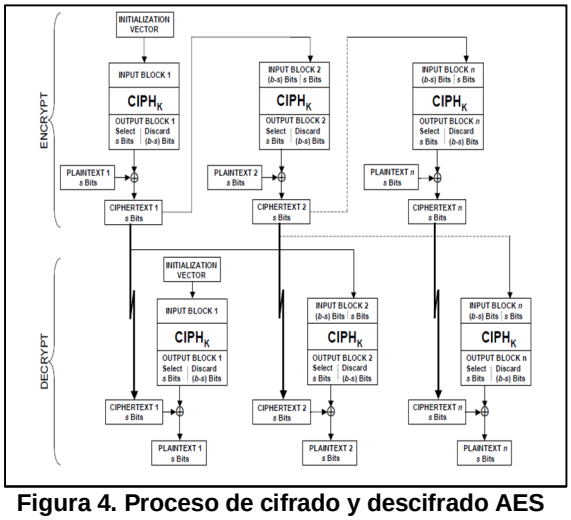

Figura 4. Proceso de cifrado y descifrado AES en modo Cifrado por Retroalimentación. [2]

2.5. Formato de mensaje SNMPv3

El formato de un mensaje SNMPv3 es muy distinto al de las dos versiones anteriores, ya que viene integrado de diferentes componentes de seguridad para la autenticación y cifrado de mensajes.

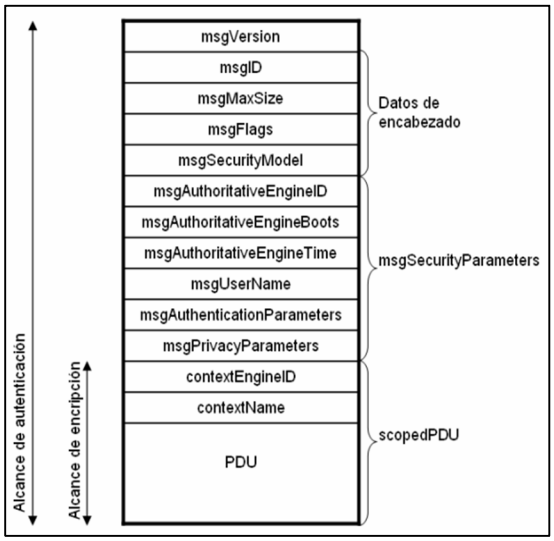

Figura 5. Formato del mensaje SNMPv3. [7]

- *msgVersion:* Indica la versión de SNMP, aunque su valor está por default en la v3.
- *msgID:* Número usado como identificador único de 32 bits que sirve para relacionar mensajes de solicitud y de respuesta.
- *msgMaxSize:* Número de 32 bits que indica el tamaño máximo en bytes que puede aceptar el emisor del mensaje.
- *msgFlags :* Cadena de 8 bits, que indica el nivel de seguridad usando solo 3 bits (menos significativos). Esta cadena contiene 3 banderas:

- reportableFlag: El valor 1 en este subcampo indica que el receptor del mensaje debe enviar de vuelta un acuse de recibo.
- privFlag: El valor 1 indica que se debe cifrar el mensaje.
- authFlag: El valor 1 indica que se debe aplicar autenticación al mensaje.

- *msgSecurityModel:* Indica el modelo de seguridad empleado por el emisor del mensaje. Esto le permite saber al receptor que modelo debe usar: SNMPv1 (1), SNMPv2 (2) y USM de SNMPv3 (3).
- *msgAuthoritativeEngineID*: Es el identificador *EngineID* de un motor SNMP autoritativo involucrado en el intercambio de mensajes.
- *msgAuthoritativeEngineBoots:* Indica el número de ocasiones que un motor SNMP reinició desde su configuración original.
- *msgAuthoritativeEngineTime:* Indica el tiempo en segundos desde que un motor SNMP incrementó *snmpEngineBoots* por última vez.
- *msgUserName:* Indica el usuario a quien se le está intercambiando el mensaje.
- *msgAuthenticationParameters:* es nulo si no se usa privacidad, de lo contrario, es un código de autenticación para el mensaje.
- *msgPrivacyParameters:* es nulo si no se usa privacidad, de lo contrario, es un valor usado para formar el IV en los algoritmos de cifrado.
- *contextEngineID:* Identifica de manera única un motor SNMP. Para mensajes provenientes, determina a que aplicación el *scopedPDU* va a ser enviado.
- *contextName:* Identifica de manera única un contexto particular dentro del contexto del motor SNMP.
- *Datos:* un PDU que debe ser de tipo SNMPv2. [7],[8]

3. Implementos y procedimientos del análisis.

Primeramente se recopila toda información concerniente a SNMP, esencialmente la versión 3, y cómo habilitarla en los dispositivos de red; así como datos sobre las aplicaciones que usaremos para realizar las pruebas de las primitivas, cómo configurarlas y su respectivo uso. En nuestro caso se usará el SNMP JManager y el WhastUp Gold v16 para el monitoreo y el Wireshark para la captura y análisis de los mensajes SNMP.

En cuanto al hardware, se utilizó: dos computadoras que tengan sus agentes SNMPv3 corriendo, una desempañará las funciones de administrador de red y la otra será monitoreada y equipos switches y routers con servicio SNMP activo de igual forma en su última versión. Con estos dispositivos, se diseña una pequeña LAN (red de área local) en la cual haremos la implementación, se debe definir si en la red hay restricciones en cuanto a subredes, una o varias VLAN (red de área local virtual) o permisos, como sucede en cualquier red corporativa real, para poder ejecutar las aplicaciones en la prueba y resolver si es que se da esta situación.

A continuación se resume en dos partes las pruebas realizadas en la red y los procesos relacionados del agente SNMPv3.

3.1. Configuración de agentes SNMPv3 y descubrimiento de los dispositivos.

Se configuran los diferentes agentes SNMPv3 (computadores, routers, switches, entre otros) y se instalan las aplicaciones de monitoreo en la computadora administradora de red; relacionado a ambas actividades hay dos procesos clave para el funcionamiento de SNMPv3: la transformación de contraseñas a llaves y la localización de las llaves generadas.

En la aplicación de monitoreo, como WhatsUp Gold, previo al descubrimiento de la red, se tienen que definir con qué credenciales (comunidades en v2 y usuarios en v3) se va a consultar los dispositivos. Al definir un usuario en el que se vayan a autenticar y cifrar los mensajes, se deben ingresar contraseñas de

autenticación y cifrado de 8 caracteres como mínimo. De igual forma, cuando se configuran los agentes de la red (switches, routers, PC, etc) con usuarios SNMPv3, se le deben definir sus contraseñas de autenticación y/o cifrado.

Cuando un usuario usa contraseñas de autenticación y/o privacidad (de mínimo 8 caracteres), el agente local las transforma en llaves de autenticación y/o privacidad mediante un algoritmo que hace uso de MD5 o SHA. Primero se forma una cadena de longitud 1’048.576 octetos al repetir el valor del password tanto como sea necesario y truncar si así se requiere al final. Esta cadena se ingresa como entrada al algoritmo hash del que hace uso el usuario para autenticar los mensajes y la salida (128 bits para MD5 o 160 bits para SHA) es la llave “maestra” o intermedia de autenticación authKey y/o privacidad privKey. [4]

Luego de esto, la llave de autenticación y/o cifrado, se localiza para cada agente específico, es decir, cada dispositivo al tener un único identificador de motor SNMP (engineID), análogo a una MAC ADDRESS, tendrá también una única llave de autenticación y/o cifrado, a pesar que dos o más agentes tengan los mismos usuarios configurados con las mismas contraseñas. Para convertir una llave maestra de usuario *Ku* a una llave localizada de usuario *Kul* en un motor autoritativo SNMP, se agrega el snmpEngineID del motor autoritativo a la llave Ku al inicio y al final de la misma, para luego aplicar una función de hash que depende del algoritmo de autenticación del que hace uso ese usuario. La salida de la función de hash (128 bits para MD5 o 160 bits para SHA) es la llave localizada de usuario Kul de autenticación y/o privacidad; las llaves intermedias nunca se almacenan o son visibles en el nodo administrado, tampoco hay forma de recuperarlas mediante alguna operación de administración. [4]

En la práctica entonces, una vez que WhatsUp Gold ha descubierto los diferentes dispositivos de red, calcula las llaves maestras de autenticación y cifrado para el o los usuarios configurados en cada dispositivo; y gracias a que en el descubrimiento se obtienen los diferentes engineID, se calculan las llaves localizadas de todos los agentes con los cuales el administrador debe comunicarse y las almacena en su base de datos interna. Por otro lado, cuando se configura un agente SNMPv3 de un dispositivo, se genera la llave maestra pero solamente la localizada es la que se utiliza para los procesos de seguridad de los mensajes y la que se almacena en el agente. A continuación se muestra a manera de ejemplo, el archivo persistente de un agente SNMPv3 de un host Ubuntu, en la Figura 6 se ve la configuración de tres usuarios y en la Figura 7, las llaves localizadas de autenticación y cifrado para cada uno, luego de reiniciar el agente.

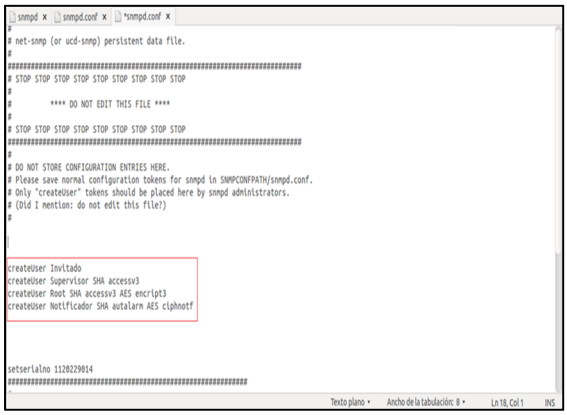

Figura 6. Configuración de usuarios y contraseñas en archivo persistente Ubuntu

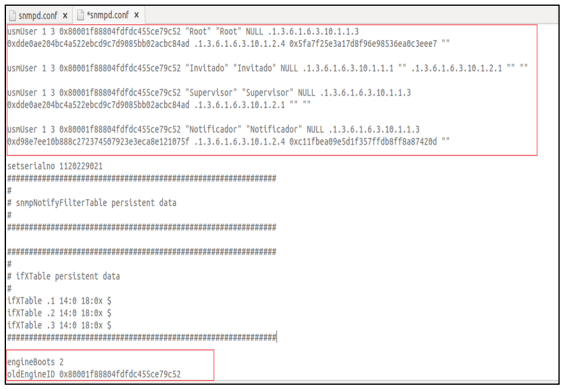

Figura 7. Llaves localizadas de usuarios en archivo persistente Ubuntu

3.2. Solicitudes a equipos monitoreados bajo versiones 2 y 3 de SNMP.

La topología utilizada para realizar las pruebas se muestra en la Figura 8.

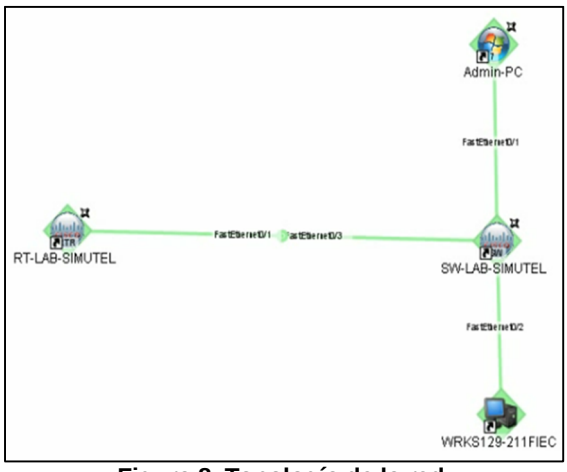

Figura 8. Topología de la red

La computadora Admin-PC tiene instalado el sistema operativo Windows 7 y desempeña la función de administrador de red, ejecuta las aplicaciones WhatsUp Gold y SNMP JManager, tiene conexión con un switch Cisco y a través de éste, con un router Cisco

2800 y una computadora Ubuntu. Debido a la reducida topología y a que el enfoque del proyecto es profundizar y facilitar el conocimiento de SNMPv3 a los usuarios, no se analizó el impacto en el rendimiento de la red al utilizar para las solicitudes de monitoreo hechas por usuarios SNMPv3 que cifren todos los mensajes. Se recomienda la realización de trabajos investigativos en el futuro, en redes corporativas reales y más grandes, para analizar los efectos de la implementación del protocolo a gran escala.

Las pruebas realizadas en la red en forma general, consisten en que el host Admin-PC monitorea la actividad del host Ubuntu y del router, y en estos últimos también fue configurado el envío de diferentes traps o alarmas hacia el host Windows, cuando ocurriera algún problema, por ejemplo, una pérdida del enlace entre los dispositivos. Todos los mensajes de solicitudes a equipos monitoreados y el envío de traps, se lo hace en las versiones 2 y 3 del protocolo, y son capturados para su análisis con el programa Wireshark. A continuación se muestra el gráfico de Utilización de CPU al host Ubuntu desde la consola de WhatsUp Gold, hecha con solicitudes SNMP de versiones 2 y 3.

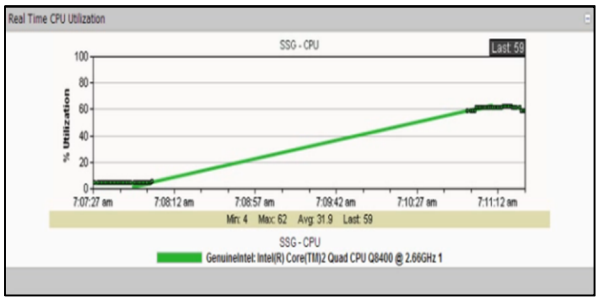

Figura 9. Utilización de CPU

No hay diferencia en los datos desplegados en el reporte en cuanto a si la solicitud fue configurada para una versión en especial, pero sí se aprecia al ver las capturas obtenidas en Wireshark, en la Figura 10 se aprecian los paquetes, en texto plano, de las solicitudes; mientras que en la Figura 11 tenemos las mismas solicitudes pero hechas con un usuario SNMPv3 que autentica y cifra los mensajes.

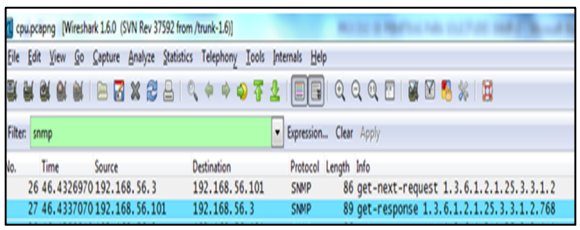

Figura 10. Mensajes SNMPv2

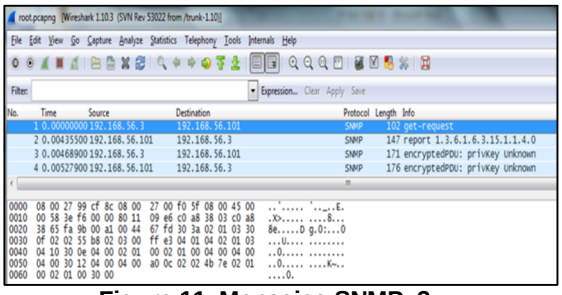

Figura 11. Mensajes SNMPv3

4. Análisis y comparación de resultados.

Se muestra la captura de Wireshark usando la comunidad de SNMPv2. Los paquetes 34 y 35 corresponden a una solicitud y respuesta respectivamente. En el get-request (Figura 12), podemos apreciar que contiene en el campo *variable bindings*, tres pares de objetos con los valores NULL para que sean respondidos en el agente de la máquina Ubuntu.

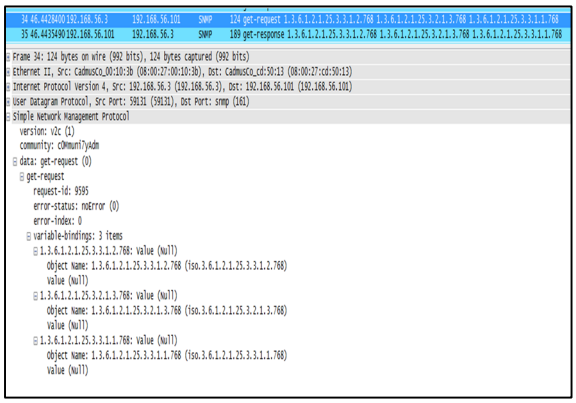

Figura 12. Paquete solicitud v2 de uso CPU

El primer objeto corresponde a *hrProcessorLoad*, que recupera el valor del porcentaje de uso del procesador en el último minuto; el segundo corresponde a *hrDeviceDescr* que describe textualmente al dispositivo, incluyendo al fabricante, revisión y posiblemente su serial; finalmente el tercer objeto es *hrProcessorFrwID* que es el ID de firmware asociado con el procesador. Todos estos OIDs tienen al final indexado un número, que indica una instancia particular para esos objetos y es un número aleatorio que el agente asigna a un procesador (si hay más de uno, se les asignarán otros números) y que permanece constante al menos, hasta el próximo reinicio del agente. A continuación se muestra el detalle del mensaje de respuesta.

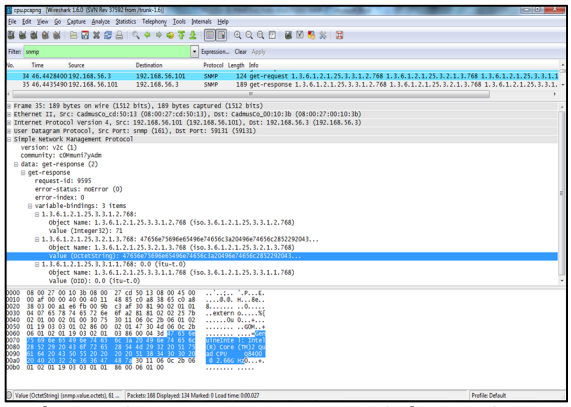

Figura 13. Paquete respuesta v2 de uso CPU

Vemos que la carga del procesador para el último minuto ha sido del 71%, que la descripción del procesador es una cadena de octetos cuyo detalle se observa en la ventana de bytes de paquete en el área sombreada, y es el mismo que aparece en la parte inferior de la Figura 9 del uso de CPU (detalles del procesador Intel); y que no hay un ID de firmware, ya que se despliega zeroDotZero (0.0) que la IETF define como un valor usado para identificadores nulos. [11],[12]

La situación cambia al usar un usuario SNMPv3 ya que ninguno de estos datos recopilados podrá ser visualizado. En WhatsUp al estar la PC vinculada con una credencial de nivel de seguridad authPriv como es el usuario Root, se logra que sus mensajes de solicitud y respuesta realicen una debida autenticación y además el cifrado de la PDU de datos. Cabe destacar que los primeros paquetes de una comunicación SNMPv3 permiten que ambos agentes se conozcan sus engineID y parámetros de tiempo para sincronizarse, antes de que la comunicación sea establecida, tras lo cual, cada agente autentica y cifra sus mensajes antes de enviarlos, de acuerdo a los procesos analizados en las secciones 2.3 y 2.4. Se analizarán los mensajes 3 y 4 (solicitud y respuesta respectivamente) a manera de ejemplo.

El mensaje 3 (Figura 14) contiene en sus campos más importantes:

- El msgVersion es 3 ya que el paquete pertenece a la versión 3.
- El msgID es 21.945 que será el mismo en el mensaje respuesta que le corresponda.
- Banderas (msgFlags). En tres bits se indican tres banderas, como vemos tiene un valor de 011 (3 en decimal) lo que indica que este mensaje tiene habilitada la autenticación, el mensaje tendrá que someterse a un proceso de autenticación en la entidad remota y además, tiene habilitado el cifrado, es decir, se cifra totalmente antes de partir hacia el destino.
- Modelo de seguridad (msgSecurityModel). Modelo de seguridad basado en usuarios (USM) vinculado a SNMPv3.
- Identificador del motor SNMP autoritativo (msgAuthoritativeEngineID). Tiene el valor 80001f88804fdfdc455ce79c52 que identifica únicamente al host Ubuntu que contiene al motor SNMP que responderá a las solicitudes (autoritativo). El número asignado por la IANA a Net-SNMP es el 8072, que se escribe en formato hexadecimal dentro de los 4 primeros octetos, lo que se observa específicamente en el tercer y cuarto octeto: 1f88. El contenido del quinto octeto, Net-SNMP Random (128) indica que del sexto al doceavo octeto se rellenará los bits con un valor aleatorio (4fdfdc45) generado por el motor SNMPv3 al momento de iniciar. [3]
- El msgUserName es Root y el msgData, que contiene las variables que son solicitadas con su valor cada una (NULL inicialmente y luego con los valores de respuesta), y que como vemos, es un PDU cifrado.

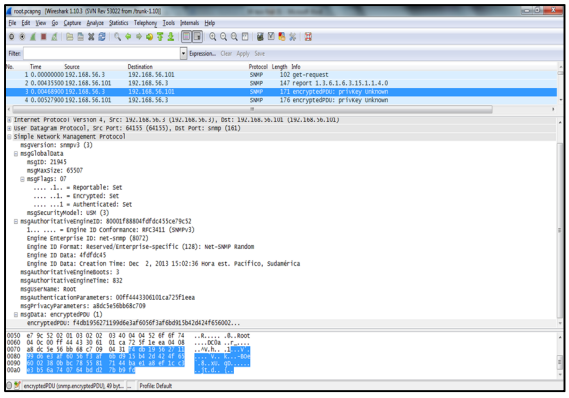

Figura 14. Paquete solicitud v3 de uso CPU

El paquete 4 (Figura 15) se comprueba es el de respuesta a la solicitud debido a que su msgID coincide con el de petición y el campo de msgData que contiene todos los OIDs (los mismos que se observaron en el paquete de versión 2) con los valores de respuesta cifrados.

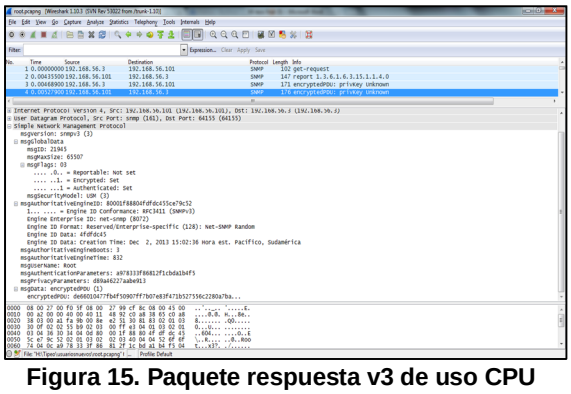

Figura 15. Paquete respuesta v3 de uso CPU

5. Conclusiones.

- Las primeras versiones de SNMP introducían seguridades muy básicas tales como los nombres

de comunidad, el identificador de los mensajes SNMP y las políticas de acceso a la MIB. Todas estas funcionalidades eran fácilmente vulnerables, porque las comunidades y datos de los mensajes viajaban en texto plano a través de la red y cualquiera podía capturar el tráfico y hacer uso de las comunidades para obtener y modificar datos de forma fraudulenta. SNMPv3 supera todas estas debilidades previas y con su esquema de usuarios provee un esquema de seguridad robusto.

- Si bien es cierto en las consolas de las aplicaciones se ingresan contraseñas para autenticar y cifrar los contenidos de los mensajes, éstas en ningún momento son visibles y a partir de ellas se calculan las llaves que en realidad son las que permiten la autenticación y cifrado de los mensajes en los agentes; y tampoco ninguna de las llaves viajan con los mensajes, sino valores producidos por los protocolos de seguridad que solamente los extremos de la comunicación entienden.
- Las llaves son localizadas para cada dispositivo mediante algoritmos seguros, lo que permite tener realmente una comunicación aislada de uno a uno, entre el equipo administrador de red y el administrado, y en el caso que uno de ellos sea vulnerado, no se perjudica a ninguno de los otros.
- El cifrado de los mensajes, es la característica más atractiva del protocolo ya que permite tener la seguridad que las operaciones administrativas no serán observadas por nadie de forma no autorizada. Solo por dar un ejemplo en el caso de AES, es prácticamente imposible recuperar la llave en el caso que se tuviera un documento original y su equivalente cifrado y el tratar de hacerlo llevaría demasiado tiempo, literalmente, siglos.
- En las pruebas de solicitudes a los agentes monitoreados, se observó que los usuarios son visibles en el paquete, a pesar que éste tenga cifrado el *scopedPDU*. Sin embargo, esto no constituye una debilidad, ya que un atacante necesitaría primero conocer la llave localizada de autenticación del dispositivo remoto, y aunque la conociera, necesitaría luego estar dentro de la ventana de tiempo del motor autoritativo para que el mensaje fuera aceptado y por último, conocer la llave de privacidad del agente. Como vemos, es mucho más complicado vulnerar la seguridad del protocolo.
- Este documento, ha demostrado que el protocolo SNMPv3 es muy sólido, confiable y seguro para las operaciones de administración de la red y su uso, pero no por eso difícil de implementar y usar. Este documento pretende facilitar la comprensión de SNMPv3 para que se implemente en las empresas.

6. Referencias.

[1] Douglas R. Mauro & Kevin J. Schmidt. (2005). *Essential SNMP 2nd Edition*. Recuperado de: http://it-ebooks.info/book/367/

[2] Morris Dworkin (Diciembre 2001). Instituto Nacional de Estándares y Tecnología. *Recommendation for Block Cipher Modes of Operation* (Seguridad de Computadoras, Estados Unidos de América) http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[3] RFC3411. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. https://datatracker.ietf.org/doc/rfc3411/?include_text=1

[4] RFC3414. User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3) https://datatracker.ietf.org/doc/rfc3414/?include_text=1

[5] RFC3826. The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User- based Security Model. https://datatracker.ietf.org/doc/rfc3826/?include_text=1

[6] Inés E. Inchauspe. (2001). Monitoreo de Redes. (Manuscrito). Universidad Nacional de Luján. Argentina. Recuperado de: http://www.tyr.unlu.edu.ar/tyr/TYR-trab/monitred/TF-Inchauspe.htm

[7] Nicolás Botero Arana. (2005). *Modelo de Gestión de Seguridad con Soporte a SNMP*. (Proyecto de Graduación, Pontificia Universidad Javeriana). Recuperado de: http://www.javeriana.edu.co/biblos/tesis/ingenieria/Tesis190.pdf

[8] Valarezo Saldarriaga y Simisterra Huila. (2011). *Implementación de un Sistema de Gestión y Administración de Redes - Basados en el Protocolo Simple de Monitoreo de Redes SNMP en la Red.* (Proyecto de Graduación). ESPOL. Recuperado de: http://www.dspace.espol.edu.ec/handle/123456789/16202

[9] William F. Ehrsam, Carl H. W. Meyer, John L. Smith, Walter L. Tuchman. (1976). Message verification and transmission error detection by block chaining, US Patent 4074066.

[10] Foro sobre criptografía. http://www.redeszone.net/2010/11/09/criptografia-algoritmos-de-autenticacion-hash/

[11] Definición de Host-Resources-MIB. Recuperado de: http://www.simpleweb.org/ietf/mibs/modules/IETF/txt/HOST-RESOURCES-MIB

[12] Ejemplo valores de Host-Resources-MIB. Repositorio. Recuperado de: http://www.observium.org/wiki/HOST-RESOURCES-MIB