5th International Conference on Information Systems and Technology Management
5º Congresso Internacional de Gestão da Tecnologia e Sistema de Informação
De 04 a 06 de Junho de 2008 - São Paulo - Brasil

PS-1003

ANTI-SPIT MECHANISM BASED ON SIP IDENTITY

Lilia Edith Aparicio (Universidad Distrital, Bogotá D.C) –medicina@udistrital.edu.co.
Octavio José Salcedo (Universidad Distrital, Bogotá D.C) -ojsalcedop@unal.edu.co
Francisco Javier Puente (Universidad Distrital, Bogotá D.C) -fjpuente2000@gmail.com

In recent years, SIP Session Initiation Protocol has become an important signaling protocol for VoIP in the convergent next generation networks NGN, given its potentiality to carry out multimedia calls from Internet, together to TCP, IP and HTTP. These three make of Internet the one who is today. However SIP is a flexible protocol, due to its capacity to extend methods and attributes. At the moment, the voice provider's services over IP VoIP networks are systems which are not open (they are not integrated to each other, E.g.: The agents MS Messenger and Yahoo! Messenger do not communicate among them). So SPIT (Spam over Internet Telephony) as not requested VoIP, has been seen as a problem not very serious today. From a reduced perspective, the SPIT is planted as a problem for knowing who will communicate with whom; the problem is confronted initially as an identity problem. If the originator identity is had, then you can authorize or not. SIP Identity is planted as a solution, given their potential to manage the identities inside SIP. JAIN SIP API is one of the most robust API's for SIP build to Java [1], so it is planted as a tool for the software development for SIP, given the Java kindness as the platform independence, mobility, among others. The main defense for not requested VoIP, are the white lists, calls studies which were labeled as SPIT, Turing Test, etc [11].

Keywords: JAIN SIP API, VoIP signaling, SIP Identity, SPAM, SPIT.

I. SIP IDENTITY

SIP como protocolo para iniciar, mantener y terminar una llamada, es muy eficiente en cuanto a la cantidad de pasos que propone, sin embargo los mecanismos existentes en SIP son inadecuados para asegurar criptográficamente la identidad del usuario final que origina las peticiones SIP, especialmente en un escenario interdominios.
Para lo cual SIP Identitiy define dos campos nuevos, para la cabecera SIP, Identity: usada para llevar la firma usada para validad la identidad, e Identitiy-Info para llevar una referencia hacia el certificado del firmante [2].

Una identidad es definida como un URI (Uniform Resourse Identifier) SIP, también llamada: dirección de registro "Address of Record AoR", empleado para alcanzar a un usuario (Ej: sip:jose@ud.com).
La especificación de SIP, no señala una forma para que el receptor de la petición SIP, verifique que el campo de la cabecera “From” ha sido correctamente diligenciada, en ausencia de algún tipo de autenticación criptográfica.

```
INVITE sip:bob@biloxi.example.org SIP/2.0
From: Alice <sip:alice@atralta.example.com>;tag=19481767
To: Bob <sip:bob@biloxi.example.org>
Call-ID: a84b4c76e66710
Cseq: 314159 INVITE
Contact: <spi:alice@pc33.atlanta.example.com>
Date: Thu, 21 Feb 2002 13:02:03 GMT
Identity:"A5oh1tSWpbmXTyXJDhaCiHjT2xR2PAwBroi5Y8tdJ+CL3ziY72N3Y+lP8eoiXlr
      ZOuwboDicF9GGxA5vw2mCTUxcoXGoKJOhpBnzoXnuPNAZdcZEWsVOQAKj/ERsYR9
      BfxNPazWmJZjGmDoFDbUNamJRjiEPOKn13uAZIcuf9zM="
Identity-Info: https://biloxi.example.org/cert
```

Figura 1. Cabecera SIP con adición de los campos Identity e Identity-Info.

Actualmente son pocos los Agentes de Usuario (AU), que soportan certificados del usuario final, necesario para autenticarse entre ellos (Ej.: S/MIME), y de igual forma, la autenticación "Digest" está limitada en cuanto a que el emisor y receptor deben compartir un secreto predeterminado, [3].

El uso de muchas aplicaciones y servicios como SIP están regulados por políticas de autorización. Estas políticas pueden ser automatizadas, o pueden ser aplicadas por humanos: tal como en el celular aparece un identificador de la llamada. En SIP "Caller-ID". Una persona puede decidir si contesta o no.

Automatizar dicha política podría ser un servicio que compare la identidad de los suscriptores potenciales contra una lista blanca "WitheList" antes de determinar si se acepta la llamada o no.

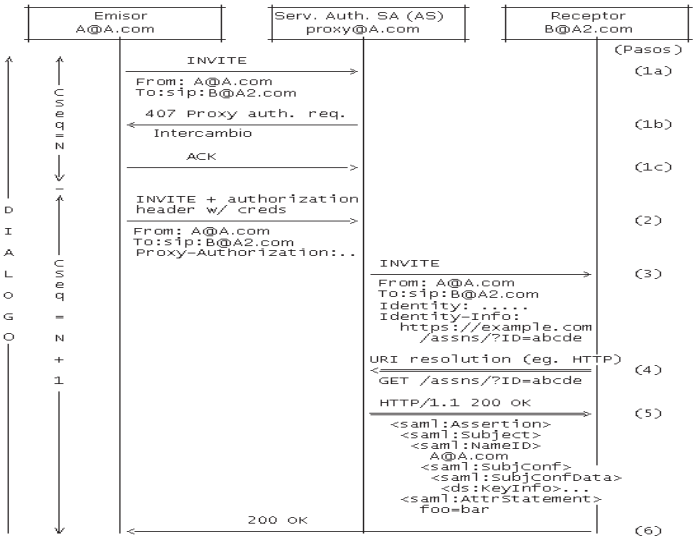

Figura 2. Servicio de Autenticación AS, SIP SAML, Perfil para obtener un atributo. Ejemplo: Una transacción INVITE.

El Servicio de autenticación SIP SAML está basado en el URI y tiene la siguiente Información requerida:

Identificación: urn: ietf:params:sip:sip-saml-profile:as:uri:attr:1.0
Como todo URN debe estar registrado con la IANA.

Información del contacto: este incluye la información del contacto.
Identificadores de los métodos de confirmación SAML: se toma de la especificación SAML 2.0.

Descripción: descripción del perfil.

A continuación se describen los pasos ilustrados en la imagen anterior

Paso1. Transacción inicial entre el emisor y SA.

Paso 2. El emisor envía un mensaje de petición SIP con credenciales de autorización al SA.

Paso 3. El SA autoriza la petición SIP y la reenvía al receptor.

Paso 4. El receptor de referencia URI SAML basado en HTTP.

Paso 5. El SA retorna una aserción SAML.

Paso 6. El receptor devuelve un 200 OK al emisor.

II COMPORTAMIENTO PARA EL SERVICIO DE AUTENTICACIÓN

1. Se debe extraer la identidad del emisor de la petición. El servicio de autenticación toma dicho valor del campo de la cabecera "From", esta AoR (Address of Record) será tomada como el campo de identidad "identitiy field". Si este campo contiene un URI SIP o SIPS, el servicio de autenticación debe extraer la porción del host del campo de identidad y compararlo con el del dominio, de lo cual el está encargado (Sección 16.4 del RFC3261). Si el servicio de autenticación no está encargado de la identidad en cuestión, este debería procesar y responder la petición normalmente, pero no debe agregar una cabecera de identidad.

2. El servicio de autenticación debe determinar si el emisor de la petición está autorizado para reclamar la identidad dada en el campo de identidad. Para llevar a cabo lo anterior, el servicio debe autenticar el emisor del mensaje. Por ejemplo: Si el servicio de autenticación es instanciado por un intermediario SIP (Proxy), este puede intercambiar la petición con una respuesta 407 usando el esquema de autenticación "Digest"[12]. Igualmente puede revisar la cabecera Proxy-Authentication enviada en la petición la cual fue enviada antes del intercambio, usando credenciales en caché. Sección 22.3 RFC3261.
Si el servicio de autenticación es instanciado por un AU SIP, se puede decir que autentica su usuario en cuanto a que el usuario puede otorgarle al AU la llave privada del dominio, o una contraseña de desbloqueo.

3. El servicio de autenticación debería asegurar que alguna cabecera de fecha "Date" en la petición, sea correcta. Políticas locales pueden dictaminar como se debe hacer la precisión de este campo, el RFC3261 recomienda una

discrepancia de máximo 10 minutos, para asegurar que la petición no sobrepase ningún verificador. Si esta cabecera contiene una hora diferente, mayor a 10 minutos de la hora actual (según el servicio de autenticación), el servicio debe rechazar la petición.
Finalmente se debe verificar que la cabecera de fecha cuadre dentro de los periodos de validez del certificado.

4. El servicio de autenticación debe hacer la firma de identidad y agregar la cabecera de Identidad a la petición que contiene la firma. Luego, se agrega la cabecera Identitiy-Info, la cual contiene un URI donde se puede adquirir el certificado. Finalmente el servicio de autenticación debe reenviar el mensaje normalmente.

III COMPORTAMIENTO DE UN VERIFICADOR

Este puede ser instanciado por un AU o un Proxy. Cuando un verificador recibe un mensaje SIP que contiene una cabecera de identidad, este debe revisar la firma para verificar la identidad del emisor del mensaje. Si dicha cabecera no esta presente en una petición y se requiere, entonces se puede enviar una respuesta 428 "Use Identitiy Header".

Para verificar la identidad el emisor del mensaje, un verificador deber llevar a cabo los siguientes pasos:

1. Debe adquirir los certificados del dominio firmante.

Dado que el certificado del dominio usado para firmar el mensaje no es conocido previamente por el receptor, las entidades SIP deberían descubrir dicho certificado por medio de la cabecera "Identitiy Info", a menos que se cuente con un servicio de búsqueda de certificados. Si el esquema del URI en la cabecera Identitiy-Info no puede referenciarse, entonces se puede enviar un mensaje 436 "Bad Identity-Info". El cliente procesa este certificado en formas usuales, incluyendo el chequeo de que no haya expirado, de que la cadena valida hacia una entidad certificadora de confianza, y que esta no aparece en listas de revocación (certificados no válidos). Una vez el certificado es adquirido, este deber ser validado usando los procedimientos definidos en el RFC3280 [4]. Si el

certificado no puede ser validado (esta auto firmado y no es confiable, o esta firmado por una entidad desconocida o no confiable, se venció o fue revocado), se debe enviar un mensaje 437 "Unsupported Certificate".

2. El verificador debe determinar si el firmante esta autorizado para el URI en el campo de la cabecera "From".

3. Se debe verificar la firma en el campo de la cabecera de identidad, siguiendo los procedimientos para generar una cadena hash. Si el verificador determina que la firma en el mensaje no corresponde con la reconstruida, entonces se envía un mensaje 438 "Invalid Identitiy Header."

4. Se debe validar las cabeceras de Fecha, Contacto y ID de la llamada (Date, Contact, Call-ID).

Debe asegurar que el valor de la cabecera Date encaje dentro del periodo de validez del certificado cuya llave privada fue usada para firmar la cabecera de identidad.

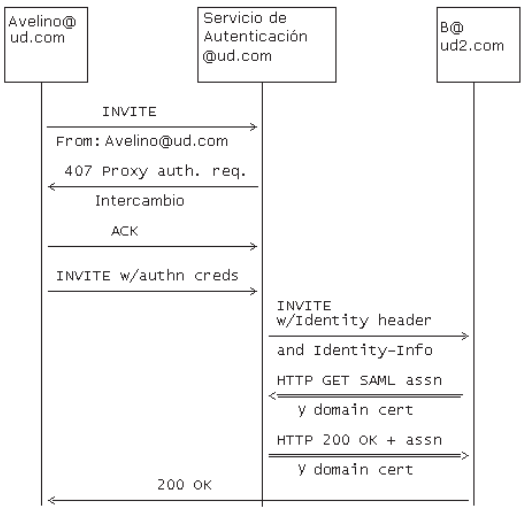

Figura 3. Flujo de la Identidad.

IV. JAVA SIP – JAIN API

Actualmente en el mercado se encuentran varias API para SIP versión 2 [3].

La API JAIN SIP soporta las funcionalidades del RFC 3261 y las siguientes extensiones; el método INFO (RFC 2976), Fiabilidad de las respuestas provisionales (RFC 3262), 'Framework' para la notificación de eventos (RFC 3265), el método UPDATE (RFC 3311), el encabezado 'Reason' (RFC 3326), el método 'Message' (RFC 3428) definido para la mensajería instantánea y el método REFER (RFC 3515) [4].

Este paquete contiene las principales interfaces que modelan la arquitectura JAIN SIP desde la vista del desarrollador de aplicaciones y del vendedor.

Vista del desarrollador:

Se trata de la implementación de la interfaz 'SipListener'. Esta interfaz define los métodos requeridos por las aplicaciones para recibir y procesar mensajes de los vendedores SIP. Un 'SipProvider' recibe mensajes de la red SIP, la cual encapsula dichos mensajes como eventos y los pasa a su 'SipListener' registrado. Una aplicación debe registrarse con 'SipProvider' para escuchar eventos dada la implementación de la interfaz 'SipListener'. Una sola interfaz 'SipListener' es obligatoria dentro de la arquitectura JAIN SIP.

Vista de los vendedores:

El vendedor implementa todas las interfaces en la especificación JAIN SIP, excluyendo la interfaz 'SIPListener', sin embargo las 2 interfaces mas importantes desde el punto de vista arquitectural son la 'SipStack' y la 'SipProvider'.

SipStack – Esta interfaz puede ser vista como una interfaz de gestión de la arquitectura JAIN SIP y solamente una puede existir por dirección IP. Esta interfaz encapsula las características de administración de SIP, tales como 'ListeningPoints' puntos de escucha que encapsulan el puerto y el transporte.

SipProvider – Esta interfaz puede ser vista como la interfaz de mensajería de la arquitectura JAIN SIP. Multiples 'SipProviders' son permitidos dentro de la arquitectura. Esta interfaz define los métodos que permiten una implementación de aplicación para que el 'SipListener' se registre con el 'SipProvider' para recibir peticiones entrantes y responderlas. Los métodos definidos para enviar mensajes SIP son también definidos dentro de la interfaz 'SipProvider' [5][6].

La implementación mas popular que se tiene de esta API, se puede encontrar en la misma página de JAIN, el proyecto se llama 'sip-comunicator'[7]. Igualmente en la página del NIST ('National Institute of Standards and Technology') se puede encontrar el proyecto NIST-SIP, con bastante documentación y algunas herramientas.

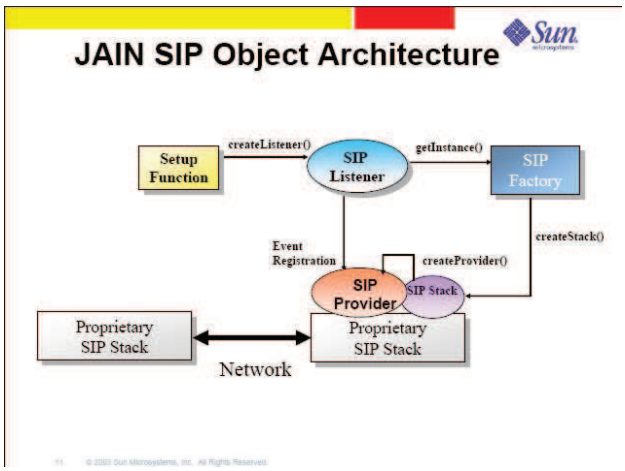

Figura 4. Arquitectura JAIN SIP

La JainSipAPI se encargaría de, básicamente dos tareas, recibir peticiones y responder a ellas. Los dos comportamientos llevados a cabo por un UAS y un UAC respectivamente. Con ello, se dependería de los siguientes puntos:

Crear la nueva cabecera.
Crear los nuevos campos para la cabecera.
Definir en que puntos sería analizada. E implementar el análisis.
Implementar la firma de las cabeceras y su resumen dentro de la nueva cabecera.
Implementar la verificación de las firmas.

Crear los atributos de la llamada.
Enviarlos dentro de la cabecera.
Analizar los datos de la llamada.
Aceptar ó rechazar la llamada.

V. DESARROLLO

Una vez configurado el entorno de programación, ya se puede iniciar el desarrollo.

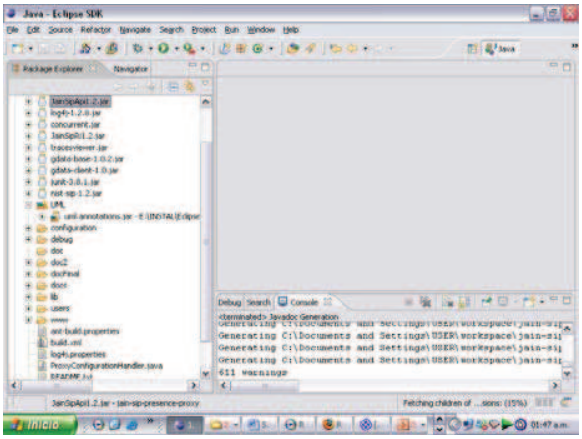

Figura 5. Desarrollando en eclipse.

1) Crear la nueva cabecera.
2) Crear los nuevos campos para la cabecera.

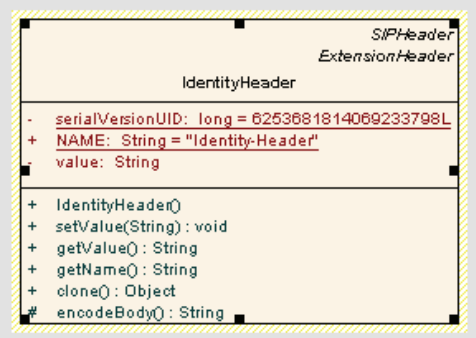

Figura 6. IdentityHeader clase.

3) Definir en que puntos sería analizada. E implementar el análisis.

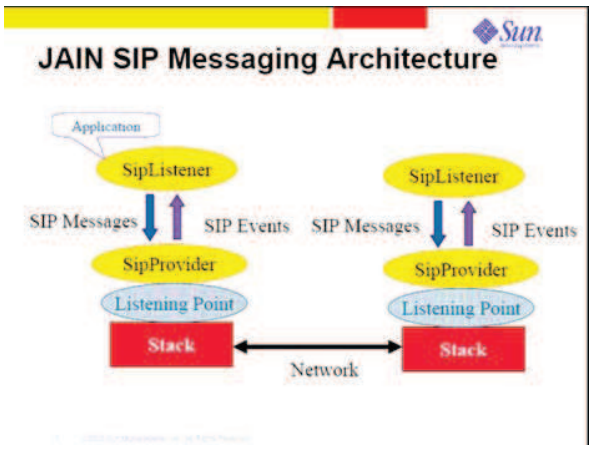

Figura 7. Arquitectura de los mensajes para JAIN SIP.

A este punto, ya se sabe a donde se va a tocar el API para que también analice la nueva extensión. En el proveedor SIP, en el evento de registro. Justo entre el 'Listener' y el Proveedor.
Por consiguiente, lo que hace falta es ubicar esto dentro del código "wrapper" implementado.

4) Implementar la firma de las cabeceras y su resumen dentro de la nueva cabecera.

En este caso se trata del evento de inicio, "sign in". Se firman las cabeceras del registro y se envían.

```
IdentityHeader identityHeaderL = new IdentityHeader();
identityHeaderL.setValue(signedHeaders);
SIPHeader identityHeader =
(SIPHeader)headerFactory.createHeader("Identity",signedHeaders);

request.setHeader(identityHeader);
SIPHeader identityHeader2 = (SIPHeader)request.getHeader("Identity");

ClientTransaction clientTransaction =
sipProvider.getNewClientTransaction(request);
clientTransaction.sendRequest();
```

Se creó un certificado digital de ejemplo. Llamado test.cer. Este fue generado usando la herramienta KEYTOOL. Para autenticar a los usuario del dominio.

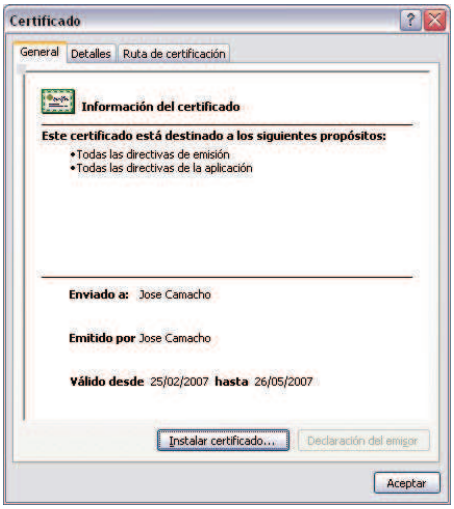

Figura 8. Certificado digital de prueba – pestaña 1.

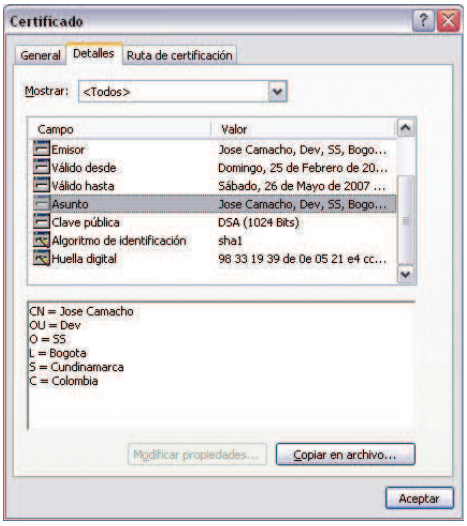

Figura 9. Certificado digital de prueba – pestaña 2.

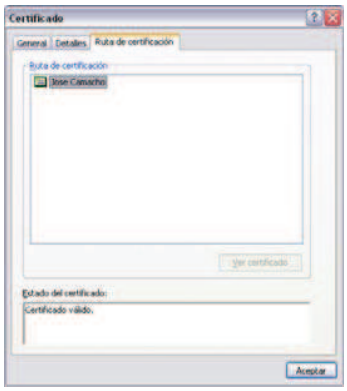

Figura 10. Certificado digital de prueba – pestaña 3.

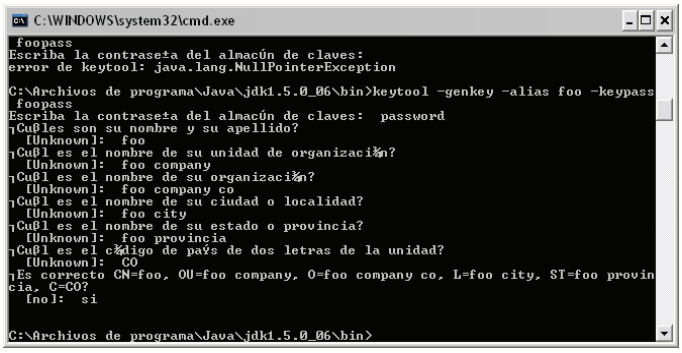

Figura 11. Creando un certificado digital con “KeyTool”.

5) Implementar la verificación de las firmas.

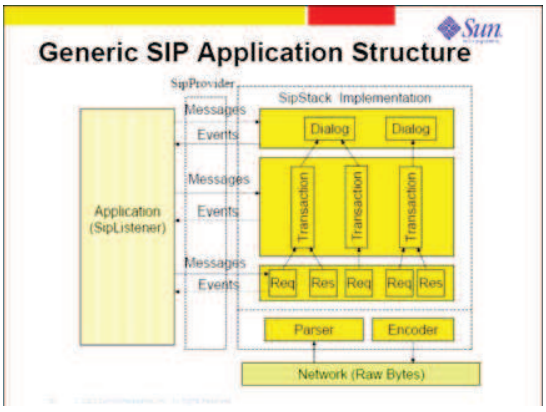

Figura 12. Estructura de una aplicación genérica SIP.

La verificación se realiza en el Proxy cuando procesa una respuesta. Al inicio puede verificar la nueva extensión y no aceptar la llamada si encuentra que las firmas no coinciden.

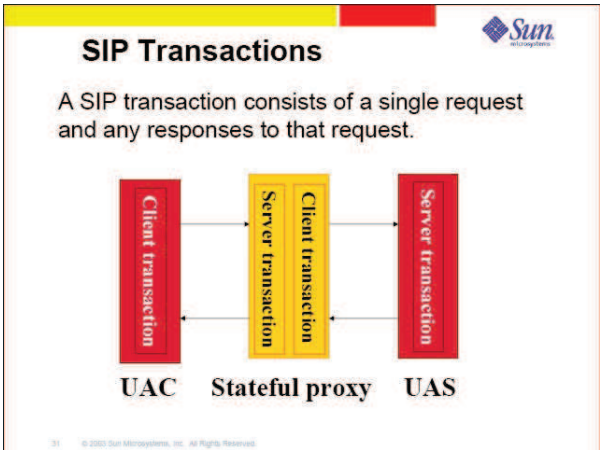

Figura 13. Transacciones SIP.

Con la verificación, se acabaría la transacción, y al agente le tocaría volver a intentar.

6) Crear los atributos de la llamada.
7) Enviarlos dentro de la cabecera.
8) Analizar los datos de la llamada.

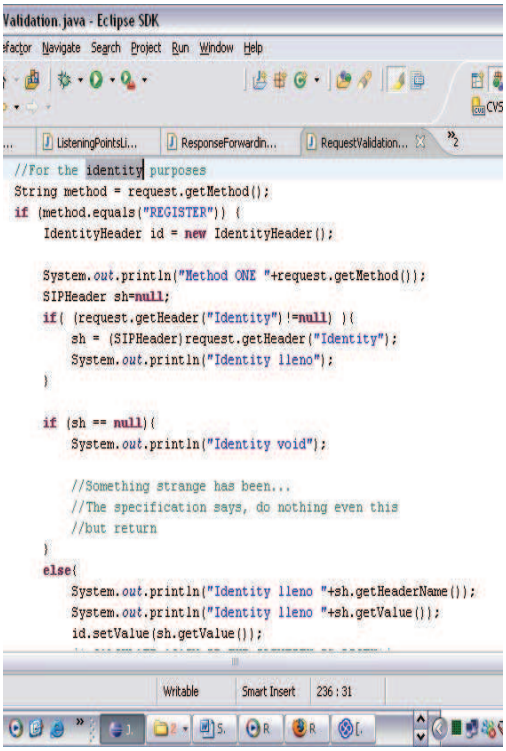

Figura 14. Validación de cabeceras.

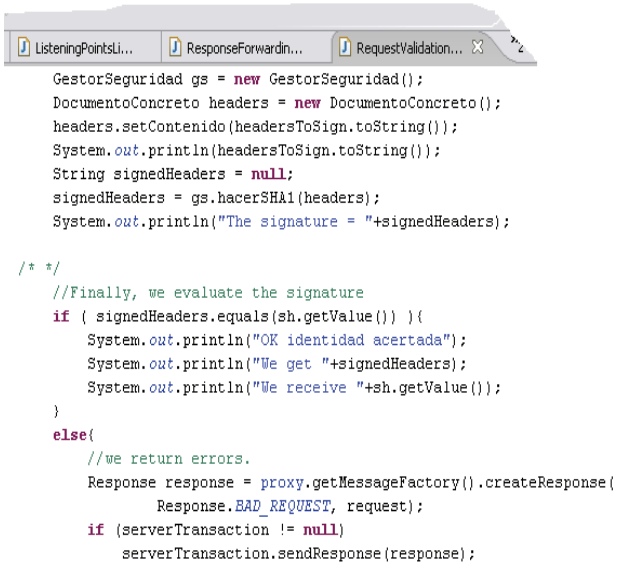

Figura 15. Se verifican las firmas. Y se toma la decisión.

9) Aceptar ó rechazar la llamada.

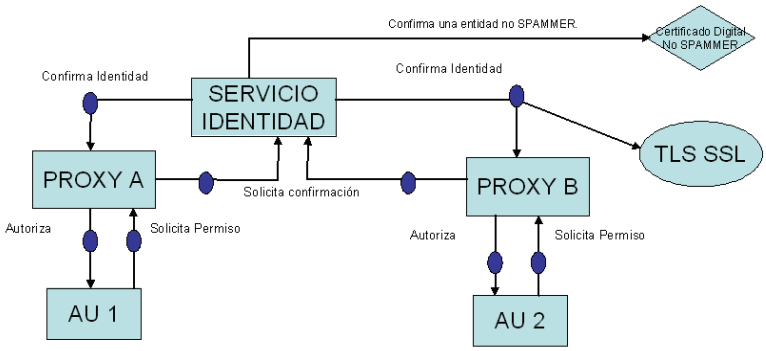

Figura 16. Resumen de la propuesta.

VI. RESULTADOS

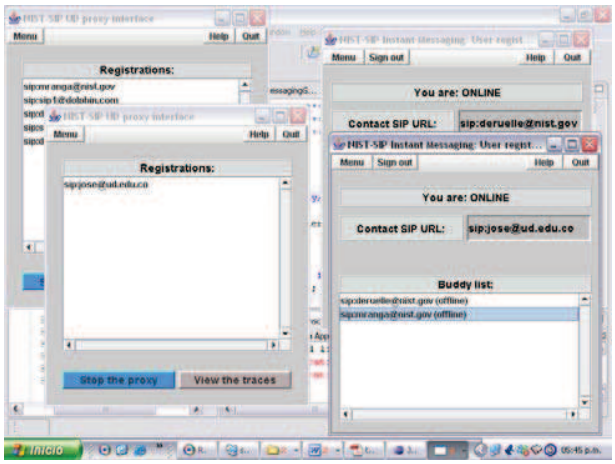

Figura 17. 2 Proxy y 2 agentes.

CARACTERÍSTICAS TÉCNICAS DE LOS EQUIPOS

Para llevar a cabo le desarrollo, se utilizó un equipo con las siguientes características:

Equipo 1.
Procesador: AMD 1800+ 1.6 GHZ 800FSB
Memoria: 1 GB.
Red: 100 Mbps. Conectado al Equipo 2 Por un cable directo.

Para llevar a cabo las pruebas, el equipo anterior estuvo a cargo de un Proxy y un agente, adicionalmente se utilizó otro equipo, encargado de uno de los agentes, con las siguientes características:

Equipo 2.
Procesador: AMD K II 500 MHZ 100FSB
Memoria: 256 MB.
Red: 100 Mbps. Conectado al Equipo 1 Por un cable directo.
Finalmente, para realizar pruebas de carga, se utilizó un tercer equipo, con otro Proxy y un agente.

Equipo 3.
Procesador: Intel Pentium 4 HT 3.2GHZ - 3.2 GHZ (Emula 2 procesadores)
Memoria: 2 GB.
Red: 100 Mbps. 1Gbps. Conectado al Equipo 1 a través de Internet. (1Mbps Equipo 3 vs 400Kbps equipo 1).

Se observó que cada agente consume en promedio 28 MB de memoria RAM, el Proxy consume la misma memoria RAM de un agente (Estos datos fueron obtenidos usando la sentencia *tasklist /FI "IMAGENAME eq java*"* la cual retorna características como el consume de memoria y CPU de cierto proceso en el SO WIN XP).
El gasto de procesador para los agentes es mínimo (10% a 15%). Incluso el Proxy fue capaz de mantener hasta 100 llamadas sin perdida de rendimiento.
El mayor gasto se presentó en la red. Es interesante que con solo IM el gasto de la red aumentara en un 20%.

El método de prueba fue básicamente a nivel de mensajes. El experimento consistió en un agente que disparó (inició) llamadas con diversas cuentas origen ficticias, durante 12 horas, hacia un conjunto de direcciones bajo la administración de un Proxy. El Proxy se mantuvo estable. Como se puede observar el consumo fue más de memoria que de CPU. Esto quizás a la cantidad de transacciones que tuvo que mantener, al igual que los registros, pues como se sabe, no se uso ninguna base de datos.

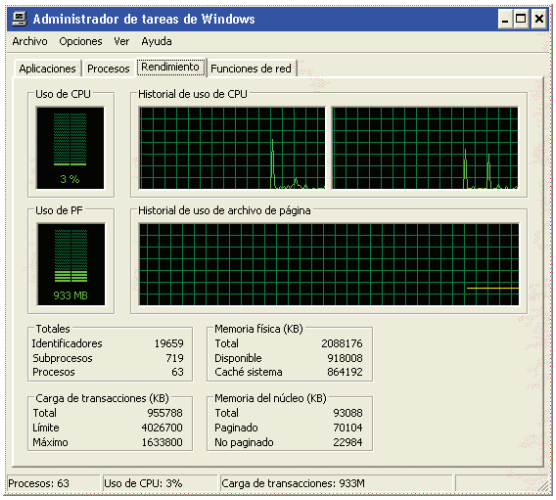

Figura 18. Gasto de memoria. En 12 días acumulado.

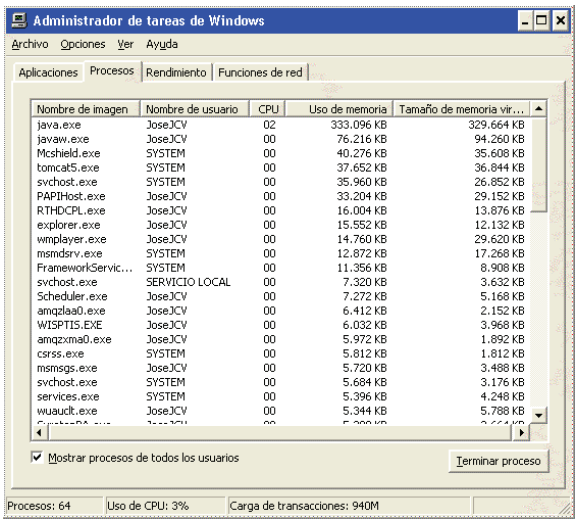

Figura 19. Gasto de memoria por parte de JVM.

El consumo de ancho de banda es muy positivo: no consume un gran ancho de banda. En promedio envía paquetes de 4,00 KB. Lo anterior se puede verificar viendo el Log del Proxy, pues este registra todos los paquetes y eventos. En las 2 horas de funcionamiento continuo llegó a 8 MB.

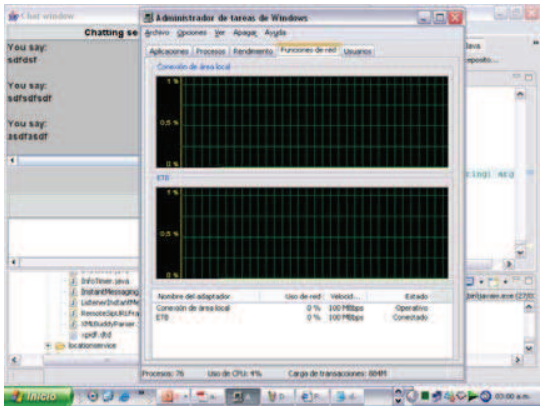

Figura 20. Consumo de ancho de banda. Se mantiene constante.

Finalmente, en cuanto a SPIT, se generó un ciclo en el agente para que dispararan llamadas con bastante frecuencia, usando diferentes atributos, válidos e inválidos. El Proxy fue muy rápido en responder, teniendo en cuenta que se usó un tan sólo el 50% de CPU de un procesador de 7000 MPI (Millones de instrucciones por segundo). El autor se atreve a decir que para un sistema mono dominio, el uso de SIP Identity realmente es muy bueno porque el Proxy conoce a sus usuarios, conoce su propio certificado y por ende SIP Identity es fuerte autenticando usuarios en un solo dominio administrativo (referencia strong authentication).

Falta esperar los resultados del estudio llevado a cabo en http://www.spitprevention.net/ [8] quienes buscan la implementación de un filtro para el SPIT teniendo en cuenta un patrón para el comportamiento del SPIT, la identidad de los usuarios, retroalimentación y uso de SAML. “Kayote Networks”, quien entre el apoyo de varios investigadores, cuenta con Hannes Tschofenig (pionero en tratar temas sociales y de seguridad para SIP, en casos de emergencia, SPIT, entre otros), y Henning Schulzrinne (creador de SIP).

VII. CONCLUSIONES

El SPIT no es ficción, como información no solicitada a través de llamadas por Internet generada por personas (SPAMMERS), que generalmente buscan de publicitar cierto producto, seguirá tomando fuerza en los próximos años. Buscar

mecanismos para controlarlo es primordial, para un óptimo desempeño y credibilidad de VoIP. Nadie usa una tecnología que le cause malestar.

No hay una única solución para el SPIT, cualquier idea es válida, pero para proveer una solución sobresaliente, se necesita una gran inversión en investigación acerca de la naturaleza 'mutante' del SPIT en cuanto a la facilidad en VoIP de cambiar de cuentas de dominio, además se conoce poco acerca del comportamiento del SPIT pues no se han presentado casos de gran envergadura.

Dada la necesidad de hacer publicidad a bajo costo y a gran escala, el SPIT se vuelve atractivo para lanzar campañas masivamente 'agresivas', volviéndose altamente entrometido, causando interrupción (incluso en el trabajo), y finalmente hasta estresante.

Desgraciadamente SIP al ser un protocolo abierto y popular, es muy susceptible al SPIT, puesto que promete acabar con monopolios, fomentar la competencia, el crecimiento de proveedores, y junto con ellos el SPIT.

Finalmente, el autor considera que SIP está entre los 3 protocolos más importantes para Internet, luego del TCP, IP y HTTP (estos tres por integrar la gran Internet que hoy día soporta millones de usuarios y aún así se mantiene gracias a la habilidad de TCP para mantener una sesión, de IP para enrutar y de HTTP para intercambio de información a nivel de aplicación), al prometer VoIP sobre Internet, ser un protocolo extensible y no propietario. Por lo tanto, vale la pena investigar sobre este, para hacerlo cada día más robusto y seguro.

VIII. BIBLIOGRAFÍA

[1] Ranga M, Helim M. Proyecto JAIN SIP.
https://jain-sip.dev.java.net/

[2] J. Peterson, C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474. http://tools.ietf.org/html/rfc4474

[3] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart. "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617. http://www.ietf.org/rfc/rfc2617.txt

[4] Rosenberg, J., "A Framework for Consent-Based Communications in the Session Initiation Protocol (SIP)" draft-ietf-sipping-consent-framework-05. www.tools.ietf.org/html/draft-ietf-sipping-consent-framework

[5] Saverio, N., "SPIT prevention state of the art and research challenges"
Third Annual VoIP Security Workshop (2006).

[6] Dean W, Keith D, Capítulo de SIP en la IETF.
http://www.ietf.org/html.charters/sip-charter.html

[7] Helim P, Ranga M. Especificación JSR-000032 JAIN SIP API. www.jcp.org/aboutJava/communityprocess/maintenance/jsr032/

[8] Comunidad Java.net, Listado de desarrollos aplicaciones desarrolladas con base en JAIN, https://www.dev.java.net/servlets/ProjectList?type=Projects&mode=TopLevel&field=ProjectName&matchType=contains&matchValue=jain

[7] Emil Ivov, Proyecto cliente VoIP para SIP, "Sip-communicator", https://sip-communicator.dev.java.net/

[9] Empresa pionera en desarrollo de soluciones contra el SPIT.
http://www.kayote.com/web/About/Research.htm

[10] Utilidad para certificados digitales y java. http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html

[11]. Jonathan Rosenberg, Cullen Jennings. The Session Initiation Protocol (SIP) and Spam.
http://www.ietf.org/internet-drafts/draft-ietf-sipping-spam-05.txt

[12] John Franks, Phillip M. Hallam-Baker, Jeffery L. Hostetler, Scott D. Lawrence, Paul J. Leach.
HTTP Authentication: Basic and Digest Access Authentication
http://tools.ietf.org/html/rfc2617