Proyecto de Grado 2006 - IM_SEC
Seguridad en Mensajería Instantánea

Informe Final

Autor: Sebastián Núñez

Tutores: Maria Eugenia Corti
Ariel Sabiguero Yawelak

Instituto de Computación

Facultad de Ingeniería

Universidad de la República

Junio de 2008

Resumen

La mensajería instantánea (IM - Instant Messaging) posibilita la comunicación entre personas mediante la utilización de aplicaciones informáticas. En el año 2005 se realizó un prototipo (IPoIM – Internet Protocol over Instant Messaging[35]) que permitió establecer un túnel IP entre dos equipos, utilizando IM como transporte. El presente trabajo analiza las consecuencias de seguridad relacionadas con el uso de la mensajería instantánea y continua el desarrollo de IPoIM. En el mismo se analizaron las herramientas de gestión de mensajería instantánea existentes en el mercado, prestando especial atención a las capacidades de filtrado de mensajería de las mismas. Se estudiaron diferentes técnicas usadas para codificar los datos que son enviados a través de la mensajería instantánea. Se analizaron las capacidades de filtrado actuales de las herramientas de gestión y se implementaron nuevas capacidades si las herramientas no lograban filtrar el tráfico generado por IPoIM. Finalmente se presenta una serie de recomendaciones de seguridad para las empresas que utilizan la mensajería instantánea como herramienta de trabajo. Estas recomendaciones se basan en los riesgos asociados al uso de la mensajería instantánea como medio de comunicación, así como los riesgos ocultos evidenciados por el uso de la mensajería como transporte para una red IP.

Palabras clave:

VPN, IP, MENSAJERIA INSTANTANEA, SEGURIDAD, COVERT CHANNELS

Índice de contenido

Indice de ilustraciones

1. Introducción

La mensajería instantánea (IM) es un servicio de comunicaciones en crecimiento. La velocidad de adopción de esta tecnología se basa en la facilidad de configuración de los clientes, gratuidad del servicio y escasa heterogeneidad en los proveedores del servicio (básicamente MSN[25], ICQ[24], Yahoo[4], AIM[23], el ya consolidado servicio Google Talk[22] y Skype[21]).

La llegada de este servicio a los puestos de trabajo empresariales no se produjo como una decisión corporativa. En la mayoría de los casos no cuenta con el soporte de la administración de tecnologías de información (TI). La mayoría de las empresas que utilizan el servicio de mensajería instantánea no consideran en profundidad los riesgos de seguridad que conlleva su uso. El principal riesgo que usualmente se identifica es la pérdida de productividad del trabajador dado que es muy difícil restringir su uso a aspectos laborales únicamente.

Como proyecto de grado en el año 2005 se desarrolló un prototipo (IPoIM - Internet Protocol over Instant Messaging[35]) que mostró que es posible utilizar la mensajería instantánea como soporte de comunicaciones para una VPN, y por ende, transmitir cualquier tipo de información a través de un servicio de IM. Información referente al proyecto antes mencionado se encuentra en [34] y en [33].

El objetivo del presente proyecto es estabilizar el prototipo construido en el proyecto de grado del año 2005, explorando los aspectos de seguridad asociados al mismo y a la vez ganar conocimiento e investigar las consecuencias que plantea en materia de seguridad la utilización de la IM para transmisión de información arbitraria. Al mismo tiempo identificar el impacto en la seguridad que tendría la posible utilización del mismo a nivel empresarial.

La industria ha comenzado a desarrollar herramientas para la gestión del uso de la mensajería instantánea. El presente trabajo evalúa las características de las mismas y analiza su capacidad de detección y filtrado del tráfico generado con este prototipo.

Otro punto que explora el proyecto actual está dado por el estudio de un método de codificación diferente al utilizado en el proyecto anterior, ya que este último solamente codificó en Base64 la información a transmitir.

El presente documento abarca los siguientes objetivos específicos y se estructura de la siguiente manera:

En el capítulo 2 se introducen los conceptos básicos sobre el funcionamiento de las aplicaciones de mensajería instantánea. Se explica el por qué de la popularidad de mensajería instantánea y analizan los beneficios que brinda su utilización, así como los riesgos asociados a la misma. En este sentido, se detallan las amenazas y las vulnerabilidades asociadas al uso, y se brindan buenas prácticas que buscan mitigar los riesgos analizados.

El capítulo 3 presenta un estudio del estado del arte de herramientas de gestión de mensajería instantánea, describe sus características y capacidades de filtrado o no del tráfico IPoIM. Para ello se analizaron sus características y se seleccionaron aquellos que potencialmente pueden filtrarlo. Se instalaron las mismas y se verificaron sus capacidades de forma empírica.

El capítulo 4 introduce los conceptos relacionados con IPoIM incluyendo: su estudio, implementación de características no desarrolladas en su versión inicial, actualización a versiones actuales de los productos de base.

El capítulo 5 describe la implementación de otro mecanismo de codificación adicional al inicial (Base64) basado en palabras de diccionario. Al mismo tiempo se detallan e implementan técnicas de detección y filtrado del tráfico generado por IPoIM.

Por último en el capítulo 6 se presentan las conclusiones finales del trabajo.

2. Seguridad en mensajería instantánea. Impacto en las empresas.

En los últimos años, la mensajería instantánea ha adquirido popularidad no sólo como medio de comunicación y de recreación para los usuarios domésticos, si no también en ambientes empresariales. Como ejemplo se citan los casos de America Online Instant Messenger[23], Microsoft MSN Messenger[25], ICQ[24], Google talk[22] y Skype[21]. Estos han cambiado la forma de comunicación entre las personas en todo nivel. El incremento del uso de la mensajería instantánea en ambietes corporativos ha sido considerable[42]. Esto se debe entre otras cosas, a que la mensajería es más económica que la telefonía, y permite mantener conversaciones con personas que se encuentran en cualquier lugar del mundo a muy bajo costo. Otros beneficios de la utilización de la mensajería instantánea se presentan más adelante en este documento.

Si bien es verdad que la mensajería instatánea brinda beneficios a quienes la utilicen, también es cierto que existen ciertas vulnerabilidades y riesgos a tener en cuenta. Cada vez más empresas se enfrentan a retos en la seguridad debido a la utilización de dichos programas de mensajería. Esto se debe fundamentalmente a que estos programas no fueron desarrollados pensando en la seguridad, si no que se pensó más en la escalabilidad. Es decir, se buscó la utilización masiva de estos programas, y no la seguridad de los mismos. Este hecho hizo que se difundan muy rápidamente, ya que muchos de ellos tienen la capacidad de configurarse para pasar a través de firewalls, lo cual hace bastatante difícil su control por parte de las organizaciones. Por ejemplo, una aplicación de mensajería instantánea puede burlar los controles establecidos por un firewall cuándo el tráfico de mensajería es transportado por protocolos bien conocidos como HTTP. Este tipo de protocolo en general está permitido en empresas, ya que se utiliza para navegación web.

El objetivo de este capítulo es introducir la mensajería instantánea y analizar los riesgos a los que se enfrentan las empresas que desean utilizar las aplicaciones de mensajería.

2.1. Funcionamiento de la mensajería instantánea

La mensajería instantánea está compuesta por uno o más servidores que brindan dicho servicio, y por los clientes que se conectan al mismo. Cada cliente utiliza una computadora con un aplicación (software) capaz de establecer una comunicación con el servidor y posibilitar el intercambio de mensajes.

Los clientes deben tener asociada una cuenta de usuario en el servidor de mensajería. Para establecer la comunicación, el cliente debe iniciar sesión en el servidor utilizando su usuario y password. Una vez establecida la conexión con el servidor, el usuario puede seleccionar a los usuarios con los cuales desea comunicarse y colocarlos en su lista de contactos. Del mismo modo, los usuarios

que quieran pueden agregar a las contrapartes de la comunicación a su lista de contactos.

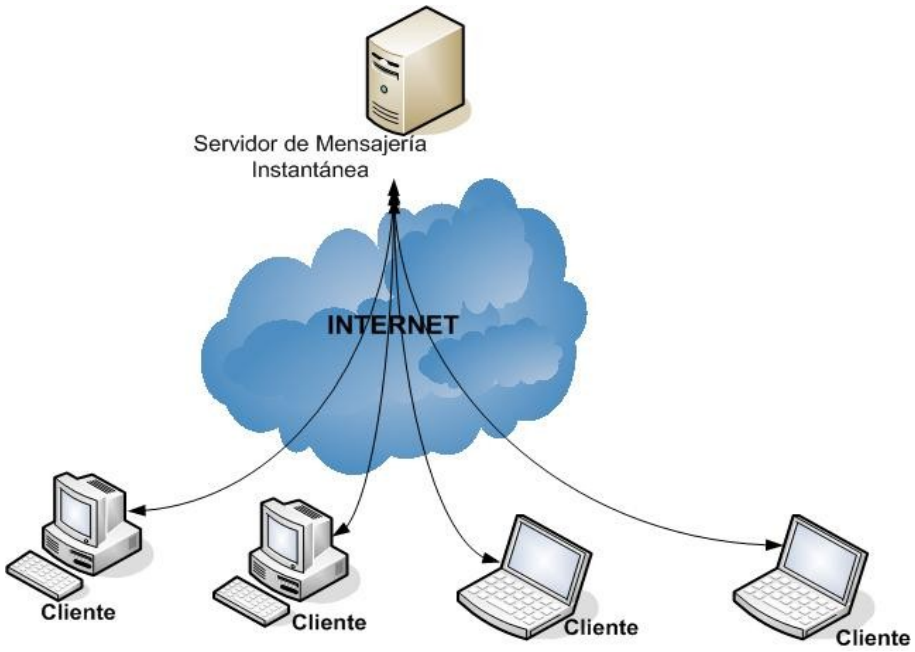

Figura 1: Funcionamiento de la mensajería instantánea

La lista de contactos cumple dos funciones básicas. Agrupa a los usuarios con los cuales se desea conversar, y también permite saber el estado de dichos contactos. Esto es, saber si están conectados al servidor o no, si su estado es "Ocupado", "Ausente", "En el telefono", etc. Esta característica se conoce como "Presencia".

Detallando un poco más el funcionamiento de la mensajería instantánea tenemos que, como muestra la figura 1:

- El servidor de mensajería brinda un servicio que acepta conexiones en uno o varios puertos determinados.
- Las aplicaciones cliente envían una solicitud de inicio de sesión (usuario y password) al servidor. El servidor notifica a los usuarios que hayan agregado en su lista de contactos al usuario que está iniciando sesión, que éste se encuentra en línea, y es posible enviarle mensajes.
- Cuando un usuario quiere establecer una comunicación con otro, lo selecciona en su lista de contactos, y le envía un mensaje. El mensaje se envía al servidor y éste se encarga de reenviarlo al destinatario.
- Algunas aplicaciones de mensajería utilizan el servidor para negociar los parámetros de la comunicación, y una vez hecho esto, la comunicación se establece entre las dos máquinas clientes utilizando algún puerto negociado anteriormente.
- Si los usarios participantes de una conversación lo desean, pueden intercambiar archivos, establecer comunicaciones de audio y video, y compartir aplicaciones en línea.

Popularidad de la mensajería instantánea.

La mensajería instantánea se volvió tan popular debido a la misma razón por la cual el correo electrónico se hizo popular. Se trata de una forma rápida y efectiva de comunicación entre personas. El uso en las empresas evita las esperas que se producen cuando el teléfono suena, la búsqueda en largas listas de extensiones, o la espera para que los operadores transfieran las llamadas. Un mensaje instantáneo es directo.

Mientras el correo electrónico tiene los mismos resultados, un mensaje instantáneo lo hace mucho más rápido. El receptor de un mensaje instantáneo ve en su pantalla el mensaje emergente y se siente más inclinado a responderlo inmediatamente. En cambio un correo puede ser respondido en otro momento.

Uno de los motivos de su popularidad es que la mensajería instantánea provee algunas características que lo hacen más personalizable y configurable. Una encuesta[42] dio estos resultados acerca del uso de las características de las aplicaciones de mensajería instantánea:

- El 39% de los encuestados usa las características de personalización (fondos de pantalla, iconos para los contactos).
- El 34% de los encuestados usa las características de mensajes de ausente.
- El 34% de los encuestados usa la mensajería para compartir fotos.
- El 29% de los encuestados usa la mensajería para compartir archivos.
- El 28% de los encuestados usa la característica de invisibilidad.

En la empresa.

Según la misma encuesta, el uso de la mensajería instantánea en las empresas se distribuye de la siguiente manera:

- Comunicarse con colegas: 58%
- Obtener respuestas rápidas en asuntos de negocios: 49%
- Comunicarse con clientes: 28%
- Intercambiar archivos: 25%
- Enviar y recibir información en una conferencia: 24%
- Enviar URLs a los colegas: 23%
- Organizar reuniones: 22%
- Usar la mensajería para conferencias de trabajo: 19%
- Organizar llamadas de conferencia: 15%
- Evitar posibles dificultades de comunicación de persona a persona: 12%

A nivel social el uso de la mensajería en la empresa se da de la siguiente manera:

- Planear actividades con la familia y amigos: 43%
- Enviar saludos por celebraciones (cumpleaños, etc.): 32%
- Compartir fotos: 26%
- Decir cosas que no se dirian en un correo electrónico: 23%
- Estar en contacto con los niños (padres trabajadores): 22%
- Mantenerse en contacto con familiares o amigos en el extranjero: 21%
- Chismear o expresar resentimientos hacia el jefe: 19%
- Coquetear: 18%
- Jugar video juegos en linea: 15%

2.2. Ventajas de utilizar mensajería instantánea.

Las ventajas que la mensajería instantánea brinda a los usuarios a nivel doméstico, así como también a nivel empresarial se citan a continuación.

- *Los programas de mensajería en su mayoría son gratuitos.*
  Por ello cualquier persona puede instalar un programa de mensajería en su computadora y de esta manera acceder a la comunicación instantánea.

- *Realización de multiconferencias.*
  No sólo es posible comunicarse con otra persona, estableciendo una comunicación uno-a-uno, si no que también se permite realizar conferencias uno-a-muchos.

- *Respuesta más rápida que al correo electrónico.*
  La comunicación al ser instantánea, es muchas veces más rápida que el correo electrónico, ya que éste último puede ser leído después, cuando el usuario receptor revise su casilla de correos en busca de nuevos mensajes. Con la mensajería instantánea, un mensaje llega de inmediato al usuario, aunque éste se encuentre realizando otra tarea en su computador.

- *Posibilidad de determinar si un usuario en línea, está disponible o no.*
  Es fácil determinar si un usuario está disponible para una conversación en línea o no. Esto se logra con los estados que permite establecer el sistema de mensajería. A esta característica se la conoce como "Presencia".

- *Reducción de costos evitando realizar llamadas telefónicas de larga distancia.*
  La disminución de costos en llamadas telefónicas es una de las principales ventajas de usar la mensajería. Sobre todo cuando de trata de contactos que se encuantran ubicados físicamente en otros paises. El costo de llamarlos telefónicamente es considerablemente mayor al costo de utilizar mensajería instantánea.

- *Posibilidad de realizar video conferencias.*
  Otra característica importante es que con muchas de las aplicaciones de mensajería existentes se permite realizar video conferencias si se dispone de una camara web.

- *Compartir archivos entre usuarios de mensajería.*
  Otro beneficio de la mensajería es que los usuarios además de establecer una comunicación, pueden si lo desean enviarse archivos mediante la mensajería, ya sea usando la transferencia de archivos, o compartiendo directamente los directorios en sus computadoras.

- *Compartir el uso de aplicaciones entre usuarios.*
  Además de compartir archivos, es posible también compartir aplicaciones entre usuarios de manera de poder trabajar sobre un mismo archivo (planilla electrónica por ejemplo)

Crecimiento de la mensajería instantánea.

Un estudio de mercado de la empresa IDC estima que para el año 2008, más de 506 millones de usuarios en el mundo utilizarán alguna herramienta de mensajería instantánea. Por otro lado The Radicati Group, realizó otro estudio de mercado en el cual prevee que para el mismo año, habrán más de 78 millones de usuarios de mensajería en las empresas.[41]

ComScore[20] que es una empresa que realiza mediciones de uso de aplicaciones en internet, presenta información sobre la segmentación del mercado la de mensajería instantánea. Dicho mercado contaba con unos 340 millones de usuarios en mayo de 2006 y los cuales se dividen entre los servicios de mensajería más populares.

La figura 2 muestra la cantidad de usuarios por sistema de mensajería en dicho año.

comsScore Media Metrix: Worldwide usage Unique Visitors (000)	
	May 2006
Total Market	339,044
IM Service	
MSN Messenger Service	203,902
Yahoo! Messenger	77,865
AIM.com/AIM App	33,952
Google Talk	3,389

Figura 2: Segmentación del mercado de IM *en Mayo de 2006[20]*

Además, la tecnología de mensajería se está haciendo popular en los dispositivos móviles, incluyendo PDAs y smart phones.

2.3. Peligro Instantáneo.

Mientras que la mensajería se vuelve popular en las organizaciones, éstas se convierten en objetivos de ataques, en general usando código maligno o técnicas de phishing[32].

Según Jon Sokoda, quien es jefe de tecnología de IMLogic, en los útimos meses, los virus de mensajería instantánea y los worms han tenido un crecimiento astronómico de 1600%, comparado con el año anterior[41].

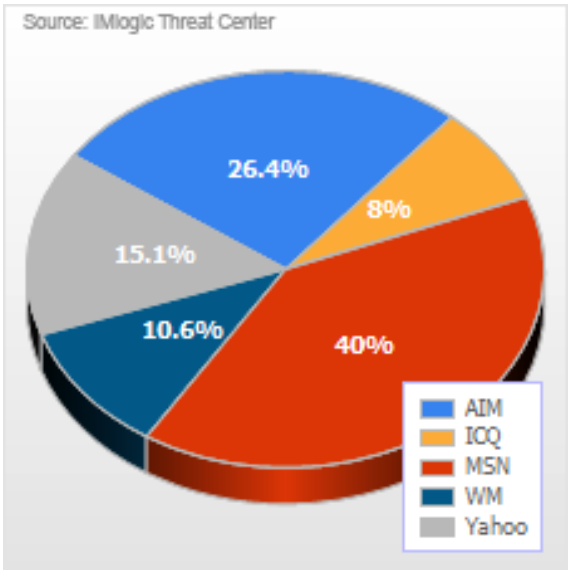

Figura 3: IM worms y su impacto en las redes públicas de mensajería instatánea

Algunos expertos en seguridad dicen que los ataques a través de la mensajería instantánea siguen los mismos patrones que siguieron los ataques a través del correo electrónico. Esto es, engaños para que las victimas seleccionen vínculos que llevan a páginas web que infectan las computadoras de los visitantes.

Lo más peligroso de estos ataques es la velocidad de propagación, debida a las capacidades de tiempo real de las aplicaciones de mensajería instantánea. Según Eric Chien, ingeniero de software de Symantec, en una simulación que la empresa llevó a cabo en 2004, se mostró que los virus instantáneos pueden propagarse a 500,000 equipos en menos de 30 segundos[41].

El centro de amenzas de IMLogic, IMlogic Threat Center, ahora llamado Symantec IM Threat Center[19] afirma que el 82% de los ataques a través de mensajería instantánea incluyen propagación de virus o worms. Otro dato importante es que, según la figura 3, el 40% de los ataques apuntan a la aplicación de mensajería

instantánea de Microsoft MSN Messenger, una de las más usadas. El 15,1% de los ataques tienen como objetivo a Yahoo Messenger. Un 26,6% afecta a los sistemas de AOL AIM y un 8% a ICQ. Estos datos corresponden al año 2005.

2.4. Riesgos: Amenazas y vulnerabilidades.

Además de sus ventajas, las aplicaciones de mensajería instantánea introducen algunos aspectos de seguridad a tener en cuenta por los usuarios y las empresas que decidan utilizarlas. De hecho, aunque las empresas no deseen utilizar la mensajería, deben hacer algunas consideraciones sobre la seguridad en lo que refiere al uso de estas aplicaciones. El no prestar especial atención en el uso que el personal de una empresa haga de las aplicaciones de mensajería instantánea puede tener un impacto negativo. No hay que olvidar que dichas aplicaciones son accesibles para cualquier empleado desde la web, y son gratuitas, por lo que cualquiera podría instalarlas sin el consentimiento de los empleadores.

2.5. Tomando conciencia de los riesgos.

El crecimiento acelerado de la utilización de la mensajería instantánea está obligando a que los usuarios y, principalmente las empresas, tomen conciencia de los riesgos a los que se enfrentan al utilizar esta tecnología. Es muy importante que conozcan cuales son las vulnerabilidades de los sistemas de mensajería y las amenazas a las que se enfrentan. El conocimiento de estos riesgos es el que asegura poder responder en tiempo y forma ante una eventualidad, que de alguna manera, ponga en peligro la confidencialidad de los datos o la imagen de una empresa.

Entendiendo los conceptos: riesgo, amenaza y vulnerabilidad.

Siempre que se habla de riesgos, se habla de vulnerabilidades y amenazas. Entonces es importante mencionar los significados que se le dan a estos conceptos. Por un lado, una amenaza es un agente externo a un sistema que podría ocasionar algun daño incidiendo en el sistema. Un ejemplo de amenaza en un sistema son los hackers. Estas personas intentan burlar la seguridad de los sistemas constituyendo una gran amenza para los mismos.

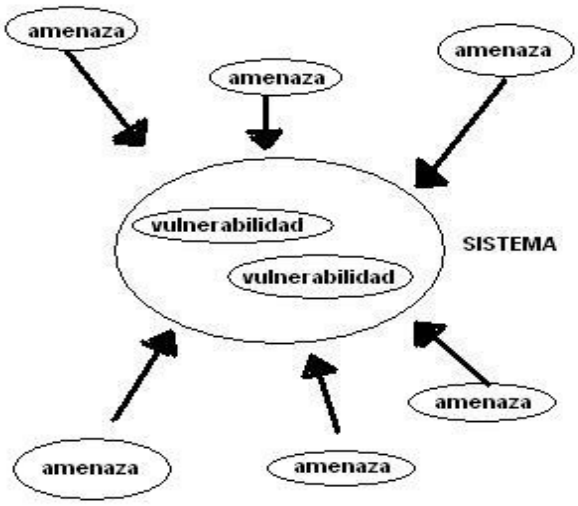

Figura 4: Amenazas y vulnerabilidades de un sistema

Las vulnerabilidades son relacionadas con los sistemas y no con el entorno. Por ejemplo, un bug en el código fuente de un programa constituye una vulnerabilidad que puede ser explotada por un hacker (amenaza del sistema). Otro ejemplo de vulnerabilidad es la posibilidad de que una funcionalidad de un programa (por ejemplo, la ejecución de scripts) sea utilizada para realizar actividades ilícitas. Por ejemplo, la mensajería instantánea permite la ejecución de código activo (scripting). Esto puede ser utilizado por un código malicioso (malware) para ejecutarse sin que el usuario lo note, y de esa manera obtener acceso a ese sistema y a otros a partir de él.

En resumen, una amenza es externa a un sistema mientras que la vulnerabilidad es interna. Una amenaza es provocada por un agente externo, mientras que una vulnerabilidad es causada por un error, una omisión o un mal diseño de una aplicación. La interacción entre amenaza y vulnerabilidad de un sistema se muestran en la figura 4.

2.6. Consecuencias en materia de seguridad del uso arbitrario de la mensajería instantánea.

Áreas de riesgo en la mensajería instantánea.

Según la organización SANS[18], las cuatro áreas de riesgo asociadas a la mensajería instantánea son:

- Malware - Worms, Virus y Troyanos transferidos a través del uso de la mensajería instantánea.

- Confidencialidad de la información – La información transferida a través de la mensajería instantánea puede ser expuesta o revelada en el proceso.
- Redes – Ataques de denegación de servicio, excesivo uso de la capacidad de utilización de una red.
- Vulnerabilidades de las aplicaciones – Las aplicaciones de mensajería instantánea contienen vulnerabilidades que pueden ser utilizadas para comprometer a los sistemas afectados.

Por otra parte, Neal Hindocha, investigador de virus para Symantec, identifica los cinco mayores problemas que amenazan la mensajería instantánea[44].

- Los gusanos y virus.
- Programas de puertas traseras o troyanos.
- El secuestro de sesiones o suplantación de personalidad.
- La denegación de servicio.
- La divulgación de información no autorizada.

Profundizando un poco más en cada uno de estos riesgos tenemos amenazas así como vulnerabilidades que es necesario mencionar.

Amenazas.

Son amenazas de un sistema de mensajería :

- HACKERS

  Si bien el término "hacker" es utilizado con diversos significados, aquí hace referencia a un usuario que por medio de la utilización de herramientas y su conocimiento, intenta burlar la seguridad de los sistemas informáticos, en busca de algún objetivo, que podría ser mal intencionado o no. En consecuencia, un hacker es una amenaza para los sistemas de mensajería instantánea. Los hackers son responsables de los ataques de tipo DoS, de robar cuentas de usuario para luego hacerse pasar por otra persona, de interceptar las conversaciones de mensajería, y de explotar los bugs que los programas de mensajería presentan en su implementación.

- WORMS

  Los worms son aplicaciones capaces de propagarse por las redes utilizando algún hueco en la seguridad de las aplicaciones. Por ello constituyen una amenaza para los sistemas de mensajería instantánea. Hasta ahora eran un problema característico de las aplicaciones de correo electrónico. Sin embargo, la mensajería instantánea provee un medio propicio para que los worms se difundan por la red. Según el jefe de investigaciones de Symantec

Eric Chien, un virus podría propagarse a medio millón de computadoras en 30 o 35 segundos usando la mensajería instantánea[43].

- TROJAN HORSES

  A diferencia de los worms, los trojan horses (aplicaciones que abren alguna puerta de entrada a un sistema) son ejecutadas por los usuarios mismos, sin saber que están facilitando la entrada a sus sistemas. Del mismo modo que los worms, los trojan horses encuentran a la mensajería como un medio propicio para llegar a más sistemas. Esto es debido a que los usuarios a menudo intercambian archivos por mesajería, y esta actividad es la que necesitan los trojan horses para llegar a ejecutarse en los sistemas víctimas. Es por eso que, en conjunto con usuarios malintencionados constituyen también una amenaza para los sistemas de mensajería.

Algunos ejemplos de código malicioso.

A continuación algunos virus y trojan horses vinculados a la mensajería instantánea:

- Assiral.A - Este worm llega como un ejecutable de windows, borra archivos y cambia la configuración de la página de inicio del Internet Explorer.
- Bizex. - El componente principal de este worm que ataca a la aplicación ICQ es la capacidad de robar información. Se propaga enviándose como un vínculo a un contacto víctima, el cual al hacer clic accede a una página web desde donde se descarga el worm.
- Bropia. - Este y sus variantes, inclyendo Kelvir y Serflog se propagan via el MSN Messenger. Este se copia al directorio de sistema de windows, descarga más malware al pc de la víctima y baja la seguridad del sistema.
- Buddypicture. - El ataque de este trojan horse, que afecta a los sistemas de AIM, comienza con un mensaje que contiene un vínculo a una página que supuestamente contiene fotografías del remitente del mensaje (que está en la lista de contactos de la víctima). El mensaje pide al usuario víctima que primero descargue un applet. Si se descarga el applet actualiza adware y spyware en la computadora víctima.
- Gabby.a. - El worm Gabby ataca los sistemas de AOL AIM e ICQ enviando vínculos a una página web. Si la víctima lo abre, se descarga spyware al PC. Este worm también abre una puerta trasera (back door) en el sistema y elimina los servicios de antivirus y de firewall de windows.
- Kelvir. - Este worm se propaga enviándose como un vínculo a un usuario de MSN Messenger con mensajes como "Hey, check this out" o "LOL, this is a funny picture of me." Los usuarios que hacen clic en el vínculo son enviados a una página web desde donde se descarga el virus.

Recientemente Kelvir dejó fuera de línea a la compañía internacional REUTERS que posee 6000 usuarios de mensajería instantánea basados en la tecnología Microsoft[40].

Vulnerabilidades.

Estas son algunas de las vulnerabilidades de los sistemas de mensajería instantánea.

- COMUNICACIONES SIN ENCRIPTAR.
  La técnica de interceptar las conversaciones realizadas por mensajería, y de esta manera acceder a datos que son compartidos entre las partes de la comunicación se denomina eavesdropping. Esto es posible ya que muchos sistemas de mensajería no utilizan encriptación en las comunicaciones, por lo que cualquiera podría interceptar dichas comunicaciones y acceder a la información intercambiada. Esto es particularmente peligroso para las organizaciones que intercambian información mediante la mensajería, ya que por ejemplo, es posible que la competencia esté accediendo a información valiosa y de esta manera esté causando pérdidas importantes a las empresas involucradas. Para realizar la intercepción de las conversaciones se pueden utilizar técnicas de sniffing[46].

- CONTRASEÑAS ALMACENADAS EN TEXTO PLANO.
  Existe una técnica llamada "account hijacking" o "robo de indentidad", que consiste en hacerse pasar por otra persona utilizando para ello los datos de su cuenta de mensajería. Una vez que un usuario malintencionado accedió a los datos de una cuenta (usuario y password), las consecuencias que esto tiene pueden ser muy serias. Dependiendo de quién sea el dueño de la cuenta de mensajería "atacada" las posibilidades de causar daño varían. No es lo mismo que se trate de la cuenta de mensajería de un usuario doméstico, a que se trate de la cuenta de un importante integrante de una compañía. De todas maneras para ambos existen riesgos. Muchas aplicaciones de mensajería almacenan las contraseñas de los usuarios en un archivo en sus computadoras. Algunas veces las claves se almacenan encriptadas, pero en otras ocasiones, las claves son almacenadas en texto plano, facilitando aún más la tarea de conseguir una clave por parte de un usuario malintencionado. Las formas de conseguir los datos de una cuenta pueden variar.

- BUGS DE IMPLEMENTACIÓN.
  Como cualquier otra aplicación, los clientes de mensajería pueden contener bugs en su implementación. Estos bugs pueden ser explotados por hackers para de esa manera obtener acceso a sistemas en donde se encuentren instalados estos clientes con bugs. Una vez que estos hackers obtuvieron acceso a sistemas, podrían instalar virus, abrir otros puertos (backdoors), etc. Esta vulnerabilidad no es exclusiva de la mensajería, pero la mensajería por medio de su lista de contactos, facilita el trabajo de los hackers ya que no necesitan escanear rangos de direcciones ip en busca de víctimas, si no que tienen a su disposición todos los contactos, y además saben cuando están conectados a la red y cuando no.

- EJECUCIÓN DE SCRIPTS: SCRIPTING.

Del mismo modo que sucede con otras aplicaciones de red, como el correo electrónico, la mensajería instantánea es un entorno muy adecuado para que aplicaciones maliciosas (malware, worms, etc) se propaguen por la red, utilizando como medio de transporte la mensajería. Esto se debe en gran medida a que muchas de las aplicaciones de mensajería permiten la ejecución de código en lenguaje de scripting. Esta característica es explotada por los virus y demás para instalarse en las máquinas víctimas, y luego propagarse a otras máquinas. La propagación la realizan utilizando las listas de contactos de los usuarios infectados, del mismo modo que los worms utilizan la libreta de direcciones de los clientes de correo electrónico. Esta vulnerabilidad crece en importancia teniendo en cuenta que cada vez más personas utilizan los programas de mensajería. Por ello, un virus podría propagarse rápidamente en millones de computadoras en muy poco tiempo si no se tienen en cuenta este tipo de vulnerabilidades.

- DENEGACION DE SERVICIO (DoS)
  Los ataques DoS son muy conocidos y no son para nada exclusivos de la mensajería instantánea, pero ésta, como cualquier otra aplicación de red, no está a salvo de este tipo de ataques. Los ataques DoS (ataques de denegación de servicio) son ataques a sistemas que proveen servicios, los cuales son saturados, impidiendo de esta manera que los usuarios accedan a dichos servicios. La forma en que son saturados es generando un flujo de información elevado, haciendo que el servidor se sobrecargue y no pueda seguir prestando servicios. Mas información sobre los ataques de este tipo se puede encontrar en [45].

- FALSA IDENTIDAD
  Una característica intrínseca de las aplicaciones de mensajería es el hecho de no poder determinar con seguridad que quien está del otro lado de la comunicación es quien dice ser.

Éstas vulnerabilidades no sólo afectan a los clientes de mensajería, si no que también afectan a los servidores. Si un usuario no autorizado obtiene acceso a un servidor de mensajería, éste podría interceptar todas las comunicaciones, hacerse pasar por cualquier usuario, realizar ataques DoS y también liberar virus para que se infecten todos los equipos.

Riesgos asociados a las empresas.

Hasta ahora hablamos de los riesgos de la utilización de la mensajería debidos a sus características técnicas, a su diseño y a su implementación. Si bien éstos son los riesgos más conocidos y son los que concentran la mayor atención para tratar de resolverlos, no son los únicos. Existen otro tipo de riesgos a los que se enfrentan las empresas. Por ejemplo, la utilización de la mensajería instantánea con fines no laborales por parte de los empleados. La popularidad de los clientes de mensajería hacen que las personas utilicen este medio para comunicarse con sus familiares y amigos. Ahora bien, el tiempo que las personas le dedican a "chatear" en su lugar de trabajo puede llegar a ser elevado. Existen varias encuestas que revelan que

muchas personas pasan más tiempo utilizando la mensajería para fines personales, que para fines laborales[42]. Esto genera grandes pérdidas a las empresas, ya que en ese tiempo, los empleados no son productivos. Otro riesgo al que se enfrentan las empresas es a la divulgación de información confidencial de la compañía a través de la mensajería. Usando las técnicas antes mencionadas, se podrían interceptar datos confidenciales en una conversación de mensajería.

Políticas de uso corporativo de la mensajería instantánea.

Gran parte de los problemas que son causados por el uso de la mensajería son debidos a que no existen políticas de uso corporativo de estas aplicaciones. Una encuesta realizada por SurfControl[39], reveló que el 90% de los encuestados tienen una política para el acceso a Internet, pero solamente un 49% tienen una política de uso de mensajería instantánea.

Muchas compañías no reconocen el daño asociado a la mensajería instantánea. Y muchas compañías que no utilizan mensajería instantánea para las comunicaciones corporativas, no son conscientes de que los empleados están usando la tecnología por su propia cuenta, ya que pueden descargar los sistemas mas populares de mensajería instantánea, en forma gratuita.

2.7. Recomendaciones de seguridad para las empresas.

Una forma de controlar la utilización de la mensajería en las empresas es utilizando firewalls. El problema con los firewalls es que no es posible impedir del todo el uso de la mensajería. Esto es debido a que, aunque se bloqueen los puertos utilizados por los clientes de mensajería, éstos intentan contactar a los servidores por medio de puertos muy conocidos como el puerto TCP/80, utilizado para navegar en la web, y de esta manera pasar inadvertido por los firewalls instalados en las empresas. Esta técnica se conoce como tunneling[47].

Buenas prácticas.

Varias organizaciones proponen buenas prácticas en lo referente a acciones a tomar para protegerse de las vulnerabilidades y del uso no autorizado de la mensajería instantánea. SANS[18] así como la empresa Symantec[31] brindan recomendaciones de seguridad. A continuación se presenta un resumen de las buenas prácticas para el uso de la mensajería instantánea.

- Establecer políticas para el uso aceptable y responsable de la mensajería y asegurarse de que todos los usuarios las conozcan y las apliquen, y que entiendan claramente los potenciales riesgos asociados.

- Restringir los privilegios de usuarios (administradores o super usuarios) solamente a los usuarios de soporte. Si un usuario necesita tener privilegios de super usuario, entonces se le debería crear una cuenta separada para el uso diario. Los usuarios comunes no deberían poder instalar programas en sus equipos.
- Asegurarse de que todos los parches para los sistemas de mensajería instantánea sean aplicados inmediatamente, así como los parches para el sistema operativo.
- Emplear software antivirus y antispam.
- No utilizar servidores de mensajería externos para uso interno. Utilizar algún proxy comercial o un servidor interno de mensajería.
- Crear canales seguros de comunicación cuando se utiliza mensajería instantánea para comunicarse con socios de negocios.
- Configurar apropiadamente los sistemas de detección y prevención de intrusos (IDS/IPS: intrusion detection/prevention system). Comprender que muchas aplicaciones de mensajería instantánea son capaces de enmascarar las comunicaciones en cualquier otro tráfico legítimo (como HTTP).
- Considerar la incorporación de productos específicamente diseñados para proveer seguridad a los sistemas de mensajería instantánea.
- Filtrar todo el tráfico HTTP a través de un proxy de autenticación para proveer capacidades adicionales de filtrado y monitoreo del tráfico de mensajería instantánea.
- Bloquear el acceso a los servidores públicos conocidos de mensajería instantánea que no han sido explícitamente autorizados. (Nota: Esto ofrece una protección parcial debido al número de potenciales servidores externos).
- Bloquear los puertos populares utilizados para mensajería instantánea. (Sólo ofrece protección parcial debido al número de protocolos existentes y sus puertos asociados, y a la habilidad de las aplicaciones de pasar por alto las restricciones de puertos).
- Monitorear usando sistemas de detección y prevención de intrusos a los usuarios que crean tuneles para mensajeria.

Microsoft por su lado, propone una lista de que se debe y que no se debe hacer al utilizar herramientas de mensajería instantánea[30].

- SE DEBE:
  - Adoptar una política de usuarios de mensajería instantánea. Los empleados deben saber que el uso de la mensajería es muy valioso para comunicarse con clientes y socios de negocios.
  - Organizar los contactos en listas separadas para los contactos de negocios, familia y amigos. De esta manera se puede evitar que un contacto social pueda ser incluído en una conversación de negocios.
  - Tener precaución de que los mensajes instantáneos pueden ser grabados. Hay que tener cuidado con lo que se dice, el mismo cuidado que se tiene con el correo electrónico.

- Prestar atención a los virus que se pueden propagar y a sus riesgos asociados.
- Mantener los mensajes instantáneos simples y saber cuando terminar una conversación.

- NO SE DEBE:
  - Usar la mensajería instantánea para enviar información sensible que puede ser vista por terceras partes. La mensajería instantánea es más apropiada para enviar información rápida como estado de proyectos, hora de reuniones, etc.
  - Permitir el uso excesivo de mensajería en el trabajo.
  - Comprometer la responsabilidad de la compañía o la propia reputación. Se pude dañar la credibilidad de la compañía.
  - Compartir información personal a través de la mensajería instantánea.
  - Confundir a los contactos con nombres de usuarios o estados engañosos.

2.8. Conclusión

Como todas las nuevas tecnologías, la mensajería instantánea presenta un entorno de nuevos riesgos, situaciones que amenazan a una organización. A su vez presenta requerimientos de nuevas habilidades para administrar las soluciones que se implementen en el caso de aceptarlas.

Como medida preventiva se deben implementar políticas de uso de mensajería en los entornos corporativos de manera de disminuir los riesgos que esta tecnología implica.

En definitiva, hay que tener cuidado con una herramienta de comunicación tan usada como los programas de mensajería instantánea, puede llevarnos a proporcionar más información de la que nos gustaría y, sobre todo, es necesario evitar que se convierta en una herramienta de distribución de virus y publicidad no solicitada.

3. Estado del arte de las herramientas de gestión de mensajería instantánea.

La introducción de la mensajería instantánea en los ambientes corporativos no ha sido debidamente planificada por los responsables de las áreas de tecnología de información. Esto se debió en gran medida a que la mensajería instantánea no fue una herramienta desarrollada para el uso corporativo. Sin embargo, muchas de las ventajas que brinda la utilización de la mensajería instantánea, han sido beneficiosas para las empresas. Es por ello, que a medida que fueron emergiendo sistemas de mensajería públicos, han sido adoptados por las organizaciones y los empleados que han reconocido el valor de estas herramientas. Fue así como paulatinamente se hizo común la utilización de servicios públicos de mensajería instantánea en las empresas y con ello, se incrementaron los riesgos en la seguridad a los que se enfrentan las mismas.

Hoy en día, se hace necesario controlar el uso de la mensajería en empresas de manera de brindar la seguridad requerida en las corporaciones. De esta necesidad, han surgido algunas herramientas capaces de administrar el uso de la mensajería instantánea. Éstas herramientas brindan la posibilidad de almacenar las comunicaciones, así como encriptar las mismas. También ofrecen interoperabilidad con redes públicas y auditoría, entre otros beneficios.

El objetivo de este capítulo es brindar una descripción de las herramientas de gestión de mensajería que actualmente están disponibles para las empresas que deseen incorporar seguridad a sus comunicaciones de mensajería.

3.1. Gestión de la mensajería instantánea.

La gestión de la mensajería instantánea abarca aspectos como:

- Administración de cuentas de usuarios.
  - Altas, bajas y modificaciones de cuentas.
- Almacenamiento de información.
  - Registro de conversaciones.
  - Estadísticas de uso.
- Bloqueo selectivo.
  - Determinar que usuarios pueden utilizar mensajería.
  - Horarios de utilización.
- Filtrado de contenido.
  - No permitir el uso de ciertas palabras.
  - Evitar las URL's en los mensajes instantáneos.
- Interconexión con redes públicas de mensajería.

Estas características se encuentran en varias aplicaciones hoy en día disponibles en el mercado. Sin embargo, no todas presentan las mismas prestaciones. Algunas se enfocan en la administración de la mensajería y otras en controlar el uso de la misma. Algunas proveen un sistema completo de mensajería para empresas sin utilizar ningún servicio público, mientras que otras se basan en servicios públicos de mensajería, e intentan dar mayor un control sobre la utilización de dichos servicios.

En este capítulo se presentan las herramientas en conjunto, dando un panorama general de las opciones disponibles para el control de la mensajería, así como la administración de la misma.

3.2. Herramientas comerciales – Appliances.

En esta sección se presentan las herramientas que brindan servicios de administración y control de mensajería utilizando para ello un dispositivo de hardware dedicado (appliance) a ese fin.

Blue Coat WebFilter

Blue Coat WebFilter[3], es una herramienta de filtrado que brinda a las empresas una protección para sus usuarios y redes de amenazas como spyware, phishing, tráfico P2P, mensajería instantánea, contenido para adultos, etc.

El funcionamiento básico consiste en filtrar el acceso a sitios web utilizando para ello una base de datos de billones de sitios restringidos.

Características

- Bloqueo y filtro HTTP
  - Bloquea/Permite tipos de archivos específicos
  - Soporte multi Lenguaje
  - Autenticación de usuarios.
- Bloqueo de spyware
- Bloqueo de mensajería instantánea (Nativa y HTTP)
- Controles básicos de IM
  - Categorización de URLs
  - Bloqueo de attachments
- Controles avanzados de IM
  - Privacidad
  - Búsqueda de palabras (keyword search)

- Registro de sesiones
- Control de aplicaciones P2P
  - Categorización de URLs
  - Direcciones IP.
- Phishing, PopUps & Banners
  - Categorización de URLs
  - Control del PopUps por aplicación.
  - Despliega información de PopUps como dirección IP y dominio para desenmascarar a los sitios de phishing.

Otros

- Licenciamiento
  - Comercial
- Versión de prueba
  - No disponible.

Akonix L7 CM5000 Appliance™ Instant IM Control.

Akonix L7 CM5000 Appliance[38] brinda una solución corporativa para la gestión de la mensajería instantánea. Protege a las redes corporativas de virus, worms y otros ataques característicos de los sistemas de mensajería instantánea. Además brinda la posibilidad de interconectarse con redes públicas de mensajería y provee mecanismos de seguridad para controlar el tráfico con dichas redes. A su vez permite almacenar todas las conversaciones establecidas de manera de llevar un registro de las mismas.

Características

1. Protege la red y a los usuarios de virus y worms que utilizan a la mensajería como transporte.
2. Previene que ciertos códigos maliciosos y/o archivos infectados sean enviados, bloqueando o escaneando las trasferencias de archivos.
3. Permite un fácil manejo de las cuentas de usuarios basándose en los permisos de usuarios.
4. Aisla a los usuarios del spam asociado a la mensajería y de otros ataques conocidos como phishing[32].
5. Asegura que la mensajería sea utilizada en concordancia con las políticas de seguridad establecidas por las organizaciones.
6. Almacena todas las conversaciones cumpliendo con ciertas normas que regulan la retención de los registros de conversaciones.

Datos técnicos.

Soporta bases de datos como:

1. MS SQL Server 2000,
2. MS SQL Server 7,
3. MS Data Engine (MSDE) (incluído).

Tiene integración con:

1. Active Directory,
2. Windows NT 4.0,
3. Sun ONE/iPlanet,
4. Directory Server (v4.1+),
5. IBM Domino,
6. Novell eDirectory.

Soporta firewalls como:

1. Check Point Firewall-1 NG,
2. Microsoft ISA Server 2000 & 2004.

Las características de archivado pueden ser provistas por:

1. iLumin Assentor,
2. Iron Mountain,
3. KVS Enterprise Vault,
4. HP RISS,
5. Legato EmailXtender,
6. ZANTAZ Digital Safe.

Soporte de Mensajería instantánea:

1. Enterprise IM Microsoft Office Live Communications Server 2003 & 2005,
2. IBM Lotus Instant Messaging,
3. Reuters,
4. Bloomberg,
5. Parlano,
6. Communicator, Inc.

Mesajería pública soportada:

1. AOL AIM,
2. AOL ICQ,
3. MSN,
4. Yahoo!.

Bloqueo peer to peer soportado:

1. BitTorrent,
2. eDonkey,
3. FastTrack,
4. Gnutella,
5. IRC,
6. Open Napster,

7. WinMX

Otros

- Licenciamiento
  - Comercial
- Versión de prueba
  - No disponible.

FaceTime IMAuditor.

FaceTime IMAuditor[29] es una herramienta desarrollada para incrementar la seguridad y la administración de las comunicaciones en tiempo real, asegurando que las comunicaciones de mensajería instantánea sean realizadas en forma segura, productiva y eficiente. Monitorea, archiva y analiza el tráfico de mensajería instantánea. Robustece la seguridad y cumple con requerimientos legales (dependiendo de los países, normas HIPAA por ejemplo) para el almacenamiento de las conversaciones de mensajería instantánea. Además soporta los clientes de mensajería más conocidos que posiblemente ya estén siendo utilizados en las organizaciones.

Características

1. Crea un perfil estándar para todos los usos de mensajería instantánea, ya sean públicas o corporativas.
2. Bloquea el spam característico de las aplicaciones de mensajería instantánea (spIM) reduciendo de esta manera la propagación de worms y virus.
3. Bloquea ataques del tipo Dia-Cero basados en mensajería instantánea.
4. Protege automáticamente de amenazas características de las aplicaciones de mensajería y P2P identificadas por los laboratorios de seguridad de FaceTime.
5. Maneja identidades con políticas de grupos y niveles de seguridad.
6. Protege contra la pérdida de información confidencial o propiedad intelectual mediante la mensajería instantánea.
7. Integra las capacidades de escaneo de archivos en busca de virus, utilizando las aplicaciones de antivirus instaladas.
8. Permite establecer límites en los tamaños de los archivos que se permiten transferir.
9. Evita la exposición de las direcciones IP impidiendo las conexiones establecidas directamente entre clientes de mensajería.
10. Almacena el 100% de las conversaciones establecidas.
11. Brinda una administración basada en una aplicación web muy intuitiva.

Datos técnicos.

Aplicaciones soportadas:

Enterprise Instant Messaging:

1. Microsoft LCS,
2. IBM Lotus Sametime,
3. Antepo,
4. Jabber,
5. Parlano MindAlign

Professional Community Networks:

1. Reuters,
2. Bloomberg,
3. Communicator Inc.,
4. PivotSolutions

Web Conferencing:

1. WebEx

Mensajería instantánea pública:

1. MSN,
2. AIM,
3. Yahoo!,
4. GoogleTalk,
5. ICQ

Requerimientos de software:

1. Microsoft Windows 2000 Server, Windows 2003 Server, o RedHat Enterprise Linux Operating System
2. Microsoft SQL Server 2000 o Oracle 9i versión 9.0.1 o 9.2

Requerimientos de hardware:

1. Pentium III 800 MHz CPU, Pentium 4 2 GHz CPU o superior
2. 1 GB de RAM

Otros

- Licenciamiento
  - Comercial
- Versión de prueba
  - Se puede solicitar una muestra del producto para evaluación. Es necesario establecer contacto con los representantes de ventas de forma que se acuerden los detalles del envío del dispositivo.

iPrism(R) Internet Filter Appliance

iPrism Internet Filter Appliance[17] es un dispositivo dedicado a la monitorización de Internet. Permite controlar el uso de aplicaciones como Instant Messaging (IM), Peer to Peer (P2P) y contenido inapropiado. Consiste en un dispositivo de hardware, un sistema operativo, y una aplicación de filtrado. Al ser un dispositivo dedicado, brinda un rendimiento superior a los otros sistemas de control de tráfico que son exclusivamente software. Existen tres opciones distintas de este dispositivo dependiendo del uso, el tamaño de la red, y los usuarios. Ellas son IPrism M500, IPrism M1200 e IPrism M3000.

Características

IPrism ofrece control de aplicaciones como:

- spyware
- malware
- mensajería instantánea
- aplicaciones P2P

Brinda una interfaz amigable para su configuración lo cual hace muy simple su utilización. Además no requiere de gran esfuerzo de mantenimiento.

Otros

- Licenciamiento
  - Comercial
- Versión de prueba
  - No disponible.

3.3. Herramientas comerciales – Software.

Hummingbird Enterprise™ IM.

Hummingbird Enterprise™ IM[37] es una solución corporativa que permite a los empleados, clientes, socios y proveedores colaboración en tiempo real, almacenamiento de conversaciones y aumento de productividad. Utiliza conexiones seguras (SSL) para transmitir mensajes, ya sean internos o hacia redes públicas. Maneja permisos de usuarios de manera de determinar qué usuarios están habilitados para participar en determinadas conversaciones.

Características

1. Comunicaciones seguras y en tiempo real con otros empleados, clientes, socios y proveedores.
2. Posibilidad de saber qué usuarios están presentes en cada contexto: miembros del equipo de un proyecto, miembros de la comunidad, propietarios de documentos, etc.
3. Captura las conversaciones en una biblioteca con un perfil completo de metadatos y permisos de acceso flexibles.
4. Convierte las conversaciones en tiempo real en hilos de conversación para su posterior revisión.
5. Comparte documentos y otros contenidos con los participantes en la conversación.
6. Transfiere archivos y comparte vínculos a los archivos de las bibliotecas de documentos, comunidades y proyectos
7. Permite un control sobre la lista personal de contactos junto con los grupos de la empresa gestionados de forma centralizada.
8. Posibilidad de integrarse con otras aplicaciones. Por ejemplo, con tareas programadas generando alertas para tareas vencidas.

Otros

- Plataforma
  - Windows
- Licenciamiento
  - Comercial.
- Versión de prueba
  - No disponible.

MS Live Communication Server 2005.

Características

Microsoft Live Communication Server 2005 brinda la posibilidad de mantener un servidor de mensajería instantánea en una organización. Puede ser extendido mediante API's de programación.

Colaboración

- Conferencias integradas
- Voz/Video/Datos Multi-Parte

Acceso de clientes

- Cliente Web
- SMS Gateway

- Clientes móbiles.

Seguridad

- Chequeo de contenido y Antivirus

Administración

- Balance de carga
- Backups

Requerimientos

- Pentium III compatible 550 MHz o superior, Dual Pentium 4 - compatible 3.0 GHz recomendado
- Microsoft Windows Server 2003 Standard, Enterprise o Datacenter Editions
- Al menos 256 MB de RAM, 2 GB de RAM recomendado
- Se recomienda discos Ultra SCSI
- Base de datos para almacenar logs.

Otros

- Plataforma
  - Windows
- Licenciamiento
  - Comercial
- Versión de prueba
  - Existe una versión de prueba disponible en la pagina de internet.

IMlogic - IM Manager for LCS 2005.

IMlogic Instant Manager para Live Communications Server es una aplicación que brinda protección de amenazas en tiempo real, aumentando las prestaciones que brinda el servidor Live Communications Server de Microsoft®. Brinda la posibilidad de implementar y administrar en forma segura la comunicación con usuarios pertenecientes a redes públicas de mensajería.

La empresa IMLogic fue adquirida por Symantec Corp. en enero de 2007, por lo que el software paso a ser propiedad de Symantec.

Características

IMlogic Instant Manager provee:

1. Control sobre usuarios externos y federados.
2. Control sobre las transferencias de archivos.
3. Escaneo de virus integrado.
4. Protección predictiva de amenazas, que detiene worms basados en mensajería instantánea antes que se propaguen.
5. Restricciones a nivel de usuario así como a nivel de grupos de usuarios.
6. Control avanzado de spIM.
7. Soporte de múltiples protocolos de mensajería.
8. Monitoreo centralizado de logs, reportes y notificaciones automáticas de eventos.

Requerimientos

Hardware:

1. 1.8GHz Pentium III
2. 30GB libre en disco
3. 512 MB RAM

Software:

1. Sistema operativo:
    1. Windows Server 2003
    2. Windows 2000 Server, Service Pack 4
2. Manejador de Bases de datos:
    1. SQL Server 2000 SP3
    2. Oracle 9iR2 (9.2.0.4)
    3. MSDE (8.00.761 incluído)
2. Requerimientos adicionales:
    1. Microsoft Core Services XML 4.0 SP2
    2. Internet Information Services 5.0 o superior
    3. Internet Explorer 6.0

Otros

- Plataforma
    - Windows
- Licenciamiento
    - Comercial
- Versión de prueba
    - No disponible ya que la empresa fue adquirida por Symantec. En su lugar Symantec ha desarrollado Symantec IM Manager.

IMlogic - IM Detector Pro.

IMlogic IM Detector Pro monitorea, escanea y bloquea una amplia variedad de sistemas públicos tales como AIM, MSN, Yahoo!, ICQ, Kazaa, Skype, entre otros. Es capaz de descubrir y bloquear selectivamente aplicaciones de mensajería instantánea y P2P. Ésta herramienta permite a las organizaciones controlar el tráfico asociado a las aplicaciones mencionadas anteriormente. Permite manejar fácilmente desde una consola todos los reportes asociados al uso de las aplicaciones, controlando de esta manera los niveles de actividad en la organización.

Características

Aplicaciones que puede monitorear y bloquear:

1. AIM (verisiones 4.7 y superior).
2. MSN (versión de protocolo MSNP8 y superior, la cual está incluída en los clientes: Windows messenger 5.0, MSN Messenger 6.0 y superiores).
3. Yahoo! (versiones 5.5 y superiores).
4. ICQ (versión ICQ Pro y ICQ Lite 2003-build 3800).
5. FastTrack (KaZaa, KaZaa Lite, Grokster, iMesh y MusicCity)
6. Gnutella (Gnutella, Morpheus, Gnucleus, Xolox, Shareaza, LimeWire, BearShare)
7. Skype (tecnología VoIP)

Con IM Detector Pro se pueden obtener las siguientes estadísticas:

1. Intentos de inicio de sesión en los sistemas.
2. Mensajes enviados.
3. Mensajes recibidos.
4. Solicitudes de transferencias de archivos realizadas.
5. Solicitudes de transferencias de archivos recibidas.
6. Direcciones IP de los usuarios.

Otros

- Plataforma
  - Windows
- Licenciamiento
  - Freeware
- Versión de prueba
  - No disponible ya que la empresa fue adquirida por Symantec.

Symantec IM Manager.

Symantec IM Manager[28] permite administrar el uso de la mensajería instantánea controlando el uso de los servicios públicos de mensajería. Posibilita el archivado de todo el tráfico de mensajería. Soporta los protocolos más conocidos como MSN Messenger, AOL Instant Messenger, Yahoo Messenger, ICQ, IBM Lotus Instant Messaging, MS Live Communication Server, Jabber, Antepo y otros.

Características

- Protege a las comunicaciones de amenazas como:
  - Virus
  - malware
  - IM spam
- Evita la exposición de información sensible, protegiendo la propiedad intelectual.
- Registra, archiva y audita las conversaciones de mensajería.
- Controles específicos sobre:
  - Trasferencias de archivos
  - Audio
  - Video
  - Aplicaciones compartidas
- Filtros de contenido
- Notificaciones y alertas en tiempo real

Requerimientos

- 1.8GHz Pentium III
- 256 RAM
- 10 GB Disco
- 600MB de memoria virtual
- Windows 2000 SP3 o Windows 2003
- MDAC 2.8 SP1 o posterior
- IE 6.0 o posterior
- XML Core Services 4.0 SP2
- IIS Web Service
- Acceso a Oracle, MS-SQL o MSDE
- IIS SMTP service (si son requeridas las notificaciones por correo electrónico)

Otros

- Plataforma
  - Windows
- Licenciamiento
  - Comercial

- Versión de prueba
  - Disponible en la pagina web.

Sun™ ONE Instant Messaging.

Sun™ ONE Instant Messaging[16] brinda mensajería en tiempo real en forma segura, así como el concepto de presencia para las empresas. Posibilita a las empresas a administrar sus propios servidores de mensajería y las cuentas de usuarios de éstos. De esta manera previene que información confidencial de la empresa sea interceptada. Ofrece múltiples mecanismos de autenticación y políticas de seguridad para los usuarios. Las comunicaciones son encriptadas utilizando SSL. Se integra con otras herramientas de Sun™ como Sun ONE Portal Server y Secure Remote Access incrementando de esta manera la seguridad.

Características

Entre otras se destacan las siguientes:

1. Administración centralizada.
2. Archivado de conversaciones.
3. Auditorías.
4. Prótección contra virus y spam.
5. Posibilidad de asociar reglas que permiten o no (deny, permit) el uso de funcionalidades que los usuarios pueden acceder.
6. Flexibilidad:
   1. Utiliza estándares como HTML y Java.
   2. Independencia de plataforma.
   3. Posee API's publicadas de manera que se pueden extender sus funcionalidades.

Requerimientos

Sistema operativo del cliente:

1. Sun Solaris 9 y 8 Operating Systems
2. Apple Mac OS 10.1
3. Microsoft Windows 98, NT, 2000, o XP
4. Red Hat Linux 7.2 o superior

Sistema operativo del servidor:

1. Sun Solaris 9 y 8 Operating Systems
2. Red Hat Linux 7.2 o superior

Requerimientos del sistema:

1. Para el servidor: Java 2 Runtime Environment, Standard Edition 1.3 (o superior)
2. Para el cliente: Java Web Start 1.0 (o superior) y Java 2 Runtime Environment, Standard Edition 1.3 (o superior)
3. Para los clientes Windows: Java Web Start 1.0 (o superior) ó Java Plug-in, y Java 2 Runtime Environment, Standard Edition 1.3 o superior
4. Memoria: mínimo 256 MB
5. Espacio en disco: 300 MB más 5 KB por usuario

Otros

- Plataforma
  - Unix, Linux
- Licenciamiento
  - Comercial
- Versión de prueba
  - Si

IM SecurityNET™.

IM SecurityNet[27] es capaz de monitorear, documentar y filtrar las conversaciones de mensajería instantánea en la red.

Características

- Protocolos soportados:
  - AIM (AOL Instant Messenger)
  - Yahoo! Messenger
  - Microsoft MSN Messenger
- Control de acceso de usuarios
  - Una de las posibilidades de esta aplicación es permitir mapear nombres de usuarios anónimos (como pueden ser las cuentas de mensajería públicas) con nombres de usuarios dentro de una organización. De esta menera se brinda la posibilidad de conocer la identidad real de los usuarios de mensajería. Ejemplo rockinredhead1895@yahoo.com = Marge Anderson.
- Notificaciones para los usuarios.
  - Con ésta aplicación es posible configurar notificaciones para usuarios o grupos de usuarios, las cuales advierten que las conversaciones están siendo monitoreadas, registradas, etc.
- Filtros de contenido.

- Además, los administradores son capaces de filtrar las conversaciones de mensajería, por medio del reconocimiento de palabras, frases, o una combinación de ellas. De esta manera se puede limitar el uso de palabras inapropiadas. Otra ventaja es la posibilidad de evitar la fuga de información de las empresas.

- Bloqueo de transferencias de archivos.
  - Permite también el bloqueo de transferencias de archivos, o en caso de no bloquear las transferencias, se puede limitar el tamaño de los archivos que se transmiten por la red.
- Escaneo de virus.
  - El escaneo de virus es otra característica de este producto.
- Reportes.
  - Por último se pueden ver un conjunto de reportes predefinidos como ser el tráfico, el ancho de banda utilizado, las transferencias de archivos realizadas y el log de los últimos 30 días de conversaciones.

Requerimientos

No tiene requerimientos específicos ya que el servicio es provisto online desde el sitio http://www.toolsthatwork.com/im_security_net.htm.

Otros

- Plataforma
  - Todas
- Licenciamiento
  - Comercial
- Versión de prueba
  - Disponible registrándose en la página web.

LivePrism IM Filter

LivePrism IM Filter[17] permite monitorear, documentar y filtrar las conversaciones de mensajería instantánea. Soporta los protocolos de mensajería más difundidos como MSN Messenger, Yahoo Messenger y AOL Instant Messenger. No necesita ser instalado ya que no está conformado por ningún hardware ni software. Consiste en un servicio on line provisto por la empresa stbernard (http://www.stbernard.com). Por este motivo puede estar operativo en pocos minutos ya que no requiere de un gran esfuerzo para su configuración.

Características

- Control de acceso de los usuarios.

- Permite el control de los usuarios ya que posibilita registrar a los usuarios y sus cuentas de mensajería públicas, junto con los datos personales de las personas. De esta manera se puede asegurar la identidad de los usuarios a los cuales se les permite el uso de la mensajería.

- Notificaciones personalizadas.
  - Se pueden configurar notificaciones para que los usuarios de mensajería tengan conocimiento de que sus conversaciones están siendo monitoreadas y/o registradas.
- Filtro de contenido.
  - Con esta característica se permite que los administradores del servicio puedan bloquear selectivamente las palabras que consideren no adecuadas. Con esto se permite filtrar el uso de palabras groseras u ofensivas, asi como evitar la salida de información sensible de las organizaciones.
- Detección y monitorización de virus.
  - La detección de virus es una característica de esta aplicación. Con ello se impide la propagación de los mismos por medio de la mensajería instantánea.
- Bloqueo de transferencias de archivos.
  - Con esta aplicación se posible bloquear el uso de transferencias de archivos, o en caso de permitirlas, se pueden monitorear de manera de saber que usuarios han transferido archivos, de que tipo, etc.

Otros

- Plataforma
  - Todas
- Licenciamiento
  - Comercial

Sigaba Secure IM

Sigaba Secure IM[2] hace énfasis en aspectos relativos a la seguridad como ser la integridad de datos, encriptación y firmas digitales para partes de mensajes o las conversaciones enteras. Se puede utilizar con Active directory o utilizar el método de autenticación incluído. La función de antivirus es provista por un plugin proprocionado por Mc Affee.

Características

- Registro completo de conversaciones
- Archivado y auditorías

- Escaneo de virus
- Monitoreo de la actividad de los usuarios
- Control de spIM

Requerimientos

- Para el cliente
  - Windows 98, NT, 2000 o XP; MacOS
- Para el servidor
  - Pentium 4 para Windows o Linux y UtraSPARC III para Sun Solaris
  - Sistema operativo Windows NT SP6, 2000 o 2003 server, Red Hat Linux 8.0 o 9.0, Sun Solaris 8 o 9

Otros

- Plataforma
  - Linux, Windows
- Licenciamiento
  - Comercial
- Versión de prueba
  - No disponible.

Effusia Business Messenger

Effusia Business Messenger[15] es un producto pensado para la pequeña y mediana empresa. Consiste en un sistema de mensajería instantánea completo, pero simple. Fue diseñado pensando en hacer fácil la tarea de instalación y mantenimiento. No requiere hardware costoso para su instalación.

Características

- Mensajes encriptados con SSL
- Control de usuarios y autenticación con administración centralizada
- Auditoría de mensajes con sofisticados reportes
- Integración con Active Directory.
- Trasferencias de archivos
- Mensajes almacenados para usuarios offline
- El servidor funciona como un servicio de Windows, y como un daemon en Linux.

Requerimientos

Effusia Business Messenger 4.1 para Windows

- Pentium II 450 MHz
- 64 MB RAM
- 50 MB libre en disco (Console)
- 100 MB libre en disco (Effusia Server)
- Service Pack 2 para Windows 2000
- Internet Explorer 5, Mozilla Firefox 1.0, Netscape 7.0 o superior (Effusia Server Management Application)

Effusia Business Messenger Server 4.0 para Linux
Distribuciones probadas por el equipo de desarrollo:

- Redhat 6.2, 7.1, 7.3, 8.0
- SuSE 8.0

Distribuciones compatibles:

- Kernel v2.2.12 con glibc v2.1.2-11 o posterior
- 32 MB RAM (64 recomendado)

Otros

- Plataforma
  - Linux, Windows
- Licenciamiento
  - Comercial
- Versión de prueba
  - Disponible en la pagina web.

WiredRed e/pop

WiredRed e/pop[14] es un sistema de gestión de mensajería instatánea para empresas que brinda la posibilidad de controlar aspectos como la integración de directorio, administración de presencia, seguridad, control de acceso y soporte para múltiples configuraciones de redes (sistemas centralizados o distribuídos). Permite habilitar/deshabilitar funcionalidades de los clientes para determinados usuarios o grupos de usuarios.

Características

- Instalación del server simple. Los clientes no necesitan otros requerimientos más que el sistema operativo.

- Integración de directorio.
- Manejo de grupos. Perfiles. Posibilidad de determinar quién puede ver el estado de los otros usuarios.
- Seguridad. Encriptación y autenticación. RC4, DES, 3DES, AES, RSA hasta 512 bits.
- Conexiones entre servidores segura.
- Auditorías. Reportes.

Requerimientos

No tiene requerimientos particulares.

Otros

- Plataforma
  - Linux, Windows
- Licenciamiento
  - Comercial
- Versión de prueba
  - Disponible en la página web.

Ipswitch Instant Messaging

Ipswitch Instant Messaging[13] es un sistema de gestión de mensajería instantánea para empresas, el cual brinda una solución para asegurar las comunicaciones a través de la mensajería.

Características

- Monitorea y registra todas las conversaciones
- Encriptación de las conversaciones utilizando el estándar 3DES.
- Integración con el Active Directory.

Requerimientos

Para el servidor:

- Microsoft Windows 2000 o 2003 Server
- Pentium 350 MHz o equivalente

- 96 MB RAM
- 100 MB libre en disco

Para el cliente:

- Windows 2000, 2003 o XP
- Pentium 350 MHz o superior
- 64 MB RAM
- 10 MB hard disk free space

Otros

- Plataforma
  - Windows
- Licenciamiento
  - Comercial
- Versión de prueba
  - Disponible en la página web.

3.4. Herramientas no comerciales.

En esta sección se presentan las herramientas disponibles para la gestión de la mensajería pero de carácter libre. Es decir, no comerciales.

Jabber.

Características

Jabber[12] es un protocolo abierto basado en el estándar XML para el intercambio en tiempo real de mensajes y presencia entre dos puntos en Internet. La principal aplicación de la tecnología Jabber es una extensible plataforma de mensajería y una red de mensajería instantánea que ofrece una funcionalidad similar a la de otros sistemas como AIM, ICQ, MSN Messenger y Yahoo.

- Es abierto -- el protocolo de Jabber es gratuito, abierto, público y comprensible. Además, existen múltiples implementaciones de código abierto para Servidores Jabber como numerosos clientes y librerías de desarrollo.
- Es extensible -- usando el potencial del lenguaje XML, cualquiera puede extender el protocolo de Jabber para una funcionalidad personalizada.
- Es descentralizado -- cualquiera puede montar su propio servidor de Jabber, además está libre de patentes y no depende de ninguna empresa de modo que se puede usar ahora y siempre con total libertad.

- Es seguro -- Cualquier servidor de Jabber puede ser aislado de la red pública Jabber, cualquier implementación del servidor usa SSL para las comunicaciones cliente-servidor y numerosos clientes soportan PGP-GPG para encriptar las comunicaciones de cliente a cliente. Además, está en desarrollo una seguridad más robusta gracias al uso de SASL y contraseñas de sesión.

Otros

- Plataforma
  - Windows, Linux
- Licenciamiento
  - GPL - Open Source
- Versión de prueba
  - Existe una diversidad de servidores que implementan Jabber diponibles para bajar. Ejemplos son Jabberd, Openfire server[1], etc.

IMSpector.

Características

IMSpector[26] es un proxy transparente con capacidad de registro y filtrado de mensajería instantánea. Actualmente puede filtrar conversaciones de mensajería utilizando para ello una lista de palabras detalladas en un archivo. Esta función esta prevista para filtrar palabras obscenas, racistas, etc.

Soporta protocolos como

- MSN
- ICQ
- AIM
- Yahoo
- IRC

Puede registrar conversaciones de diversas modos:

- Registro hacia archivos
- Registro hacia bases de datos
  - MySQL
  - SQLite

Corre en Linux y en BSD. Fue desarrollado bajo la licencia GPL. Esta escrito en C++.

Otros

- Plataforma
  - Linux
- Licenciamiento
  - GPL - Open Source
- Versión de prueba
  - Se puede bajar el software.

IM-Filter

IM-Filter[11] es un servicio (daemon) para LINUX, escrito en C. El mismo permite analizar y filtrar el protocolo ICQ. Se ejecuta en el espacio del usuario y lee paquetes de ICQ por medio de la librería `libnetfilter_queue`.

El propósito principal de IM-Filter es ejecutarse como un gateway. De esta manera puede interceptar todo el tráfico de ICQ de la LAN. De todas maneras, también puede ejecutarse directamente en las computadoras de usuarios finales.

El diagrama presentado en la figura 5 muestra como se integra el servicio en un sistema GNU/Linux.

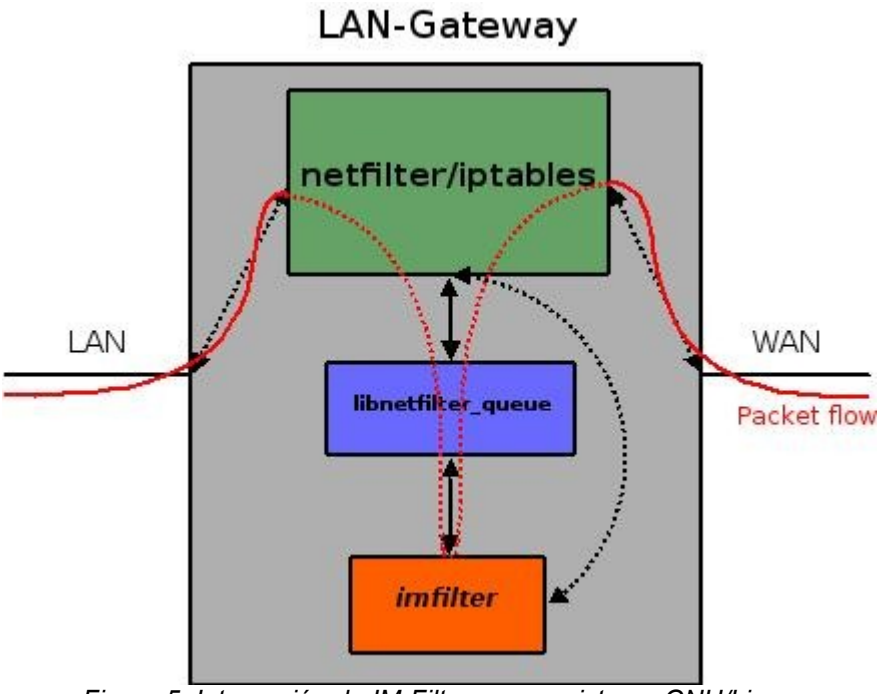

Figura 5: Integración de IM-Filter con un sistema GNU/Linux

Un aspecto destacable es que fue diseñado modularmente. Por este motivo es

posible desarrollar y agregar fácilmente plugins para otros protocolos como MSN o IRC. Actualmente se distribuye con un plugin para IRC.

Junto con el código fuente se proveen scripts adecuados para iptables, de modo que solamente los paquetes de ICQ sean considerados en la cola de `netfilter`.

Características

Registro de mensajes:

IM-Filter es capaz de identificar y registrar mensajes que son enviados a través del protocolo ICQ. El registro consiste en un timestamp, la dirección IP, el número de usuario de ICQ (UIN) y el mensaje. Un ejemplo del registro sería:

```
[2007-07-27 11:24:21] IP "10.0.0.1" sent following message to
#123456789: "hello, this is a test"
```

Lista negra por mensajes y por números UINs:

IM-Filter maneja dos "balcklists". Por mensajes y por UINs. La lista por mensajes almacena cadenas de texto que no pueden ocurrir en los mensajes de texto. La segunda lista contiene los números únicos de los usuarios que no pueden conectarse a la red ICQ. Ambas listas pueden ser re leídas en tiempo de ejecución por medio de señales que son enviadas.

Detección y bloqueo de transferencias de archivos:

Las transferencias de archivos pueden ser bloqueadas invocando al programa con esa opción en la línea de comandos. De esta manera quedan bloqueadas todas las transferencias de archivos entre usuarios. Además de ser bloqueadas, cada intento de transferencia es registrado, y aparece de la siguiente manera:

```
[2007-07-06 17:01:26] FILTRANSFER detected (outgoing): receiver
uin: #123456789 || filename: test.txt || size: 143360B
```

Lista de usuarios conectados actualmente:

Esta lista permite saber en todo momento que usuarios están conectados. Se trata de un archivo de texto con esa información almacenado por defecto en /var/log/imfilter.users. Cada entrada del archivo contiene la dirección IP y el número único UIN de cada usuario conectado. Un ejemplo es:

UIN=123456789, IP=10.0.0.1
UIN=234567890, IP=10.0.0.22
UIN=345678901, IP=10.0.0.4

Otros

- Plataforma
  - Linux
- Licenciamiento
  - GPL – Open Source
- Versión de prueba

- Se puede bajar el software desde el website.

3.5. Otros recursos para controlar la mensajería instantánea.

A continuación se presentan herramientas que brindan alguna posibilidad de control sobre la mensajería instantánea, pero que no fueron desarrolladas exclusivamente para este fin. No se consideran herramientas de "gestión" de mensajería por carecer de características necesarias como ser, administración de usuarios, almacenamiento de conversaciones, etc.

Cisco IOS®.

Cisco IOS® es un sistema operativo propietario de Cisco Systems. El mismo se desarrolló como software base para dispositivos específicos con funciones de red (routers, switches, firewalls, etc.). En su versión Cisco IOS® Software Release 12.3(14)T, se incluyen características que permiten controlar tráfico específico de determinadas aplicaciones[36]. En particular, mensajería instantánea.

Características.

Cisco IOS® Software Release 12.3(14)T permite controlar el tráfico saliente de una red privada, por medio de dos características a saber:

1. Statefull inspection:

   Permite monitorear el tráfico de red saliente en un firewall y de ésta manera, anticipar el tráfico entrante que debe permitir desde una red pública. Esto es, en base a las conexiones que los usuarios establecen desde dentro de la red privada, se permiten conexiones entrantes en respuesta a las solicitudes requeridas. Por ejemplo, si un usuario desea realizar una transferencia de archivos utilizando el protocolo FTP, debe iniciar una conexión saliente. Cisco IOS® mediante la característica de statefull inspection, monitorea la iniciación, el establecimiento y la finalización de las conexiones de red, y permite en base a esto las conexiones entrantes. Por otro lado, utiliza listas de control de acceso para bloquear conexiones no deseadas desde las redes públicas.

2. Application inspection:

   Esta característica permite escanear el tráfico de aplicaciones específicas en busca de tráfico embebido en otro tipo de tráfico de red conocido. Por ejemplo, muchas aplicaciones de mensajería instantánea y P2P utilizan técnicas para embeber en protocolos conocidos como HTTP, el tráfico generado de manera de pasar inadvertidos por firewalls. Esto es común ya que en la mayoría de las organizaciones se permite la salida de tráfico HTTP hacia redes públicas. Haciendo uso de ello, aplicaciones como MSN

Messenger, KaZaA, etc. se abren paso a través de los firewalls corporativos. Application inspection permite controlar que el tráfico saliente HTTP sea precisamente trafico HTTP y que no se trate de una aplicación utilizando a éste protocolo como túnel de red.

Estas características permiten a un administrador de red, permitir o no el uso de mensajería instantánea. Si bien esto no representa una gestión de la mensajería instantánea, es una característica a tener en cuenta si se desea impedir el uso de dichas aplicaciones, sin perjudicar el uso del resto de la aplicaciones como los servicios web.

3.6. Comparativa de herramientas.

Cuadro comparativo.

	Blue Coat WebFilter	Akonix L7 CM5000 Appliance ™	FaceTime IMAuditor	iPrism(R) Internet Filter Appliance	Hummingbird Enterprise™ IM	MS Live Communication Server 2005	IMlogic - IM Manager for LCS 2005	IMlogic - IM Detector Pro
Escaneo y/o bloqueo de transferencias de archivos		Si	Si			Si	Si	
Administracion de cuentas de usuarios		Si			Si	Si	Si	
Bloqueo de spIM	Si	Si	Si			Si	Si	
Almecenamiento de conversaciones	Si	Si	Si	Si	Si	Si		
Administracion por aplicación web			Si			Si		Si
Evita la exposicion de direcciones ip			Si					
Soporte de multiples protocolos de mensajeria							Si	
Filtrado de mensajeria instantanea	Si			Si		Si		Si
Independencia de plataforma				Si				
API de programacion						Si		
Busqueda de palabras (keyword search)	Si							
	Symantec IM Manager	Sun™ ONE Instant Messaging	IM SecurityNET ™	LivePrism IM Filter	Sigaba Secure IM	Effusia Business Messenger	WiredRed e/pop	Ipswitch Instant Messaging
Escaneo y/o bloqueo de transferencias de archivos	Si	Si	Si	Si				
Administracion de cuentas de usuarios	Si	Si						
Bloqueo de spIM	Si	Si	Si	Si	Si			
Almecenamiento de conversaciones	Si	Si	Si	Si	Si	Si	Si	Si
Administracion por aplicación web	Si							
Evita la exposicion de direcciones ip								
Soporte de multiples protocolos de mensajeria	Si		Si	Si	Si			
Filtrado de mensajeria instantanea	Si	Si	Si	Si				
Independencia de plataforma		Si	Si	Si		Si	Si	
API de programacion		Si						
Busqueda de palabras (keyword search)	Si		Si	Si				
	Jabber	IMSpector	IM-Filter	Cisco IOS				
Escaneo y/o bloqueo de transferencias de archivos			Si					
Administracion de cuentas de usuarios	Si							
Bloqueo de spIM	Si							
Almecenamiento de conversaciones	Si	Si	Si					
Administracion por aplicación web	Si							
Evita la exposicion de direcciones ip								
Soporte de multiples protocolos de mensajeria	Si	Si						
Filtrado de mensajeria instantanea			Si	Si				
Independencia de plataforma	Si							
API de programacion	Si	Si	Si					
Busqueda de palabras (keyword search)		Si	Si					

Figura 6: Cuadro comparativo de herramientas de mensajería instantánea

La figura 6 muestra un cuadro comparativo que resume las principales características de las herramientas presentadas anteriormente.

En el cuadro anterior sobresalen debido a sus características, las siguientes herramientas de gestión de mensajería:

1. Microsoft Live Communication Server 2005
2. Symantec IM Manager
3. IM Security Net
4. LivePrism IM Filter
5. IMSpector
6. Jabber

Las mismas poseen características que brindan la posibilidad de investigar con ellas. Las primeras cuatro son herramientas comerciales, pero las dos primeras corresponden a aplicaciones que el usuario debe instalar en un servidor, mientras que las otras dos, son servicios provistos por empresas, que no requieren de una instalación local en un servidor propio. Las últimas dos son aplicaciones no comerciales (free/open source). Todas permiten evaluarlas mediante versiones de evaluación. Por ello se tomó un representante de cada categoría para realizar pruebas con IPoIM. Como herramienta comercial se tomó Symantec IM Manager, ya que permite utilizar cuentas de mensajería públicas, y provee mecanismos de filtrado de palabras. De las aplicaciones comerciales que se prestan como un servicio online se utilizó IM SecurityNet, ya que ésta empresa brindaba un servicio de prueba por 30 días. Por último se utilizó IMSpector como herramienta no comercial debido a su implementación simple y su potencial de expansión. De manera que es altamente recomendable realizar pruebas de filtrado con éstas herramientas. IMSpector por su lado, provee un módulo de filtrado con posibilidades de extensión que lo hace un candidato muy interesante para realizar pruebas.

3.7. Conclusión

Existe una variada oferta de aplicaciones de gestión de mensajería instantánea. Con estas se ofrece mayor seguridad a las empresas que decidan optar por una solución de mensajería como herramienta de trabajo. Sin embargo, debido a que el uso de la mensajería tiene asociado el incremento de los problemas de seguridad, conviene además de utilizar alguna herramienta de gestión, seguir las buenas prácticas proporcionadas por algunas empresas que se dedican a estudiar los problemas de seguridad asociados a estas aplicaciones. Estas buenas prácticas fueron presentadas en la sección 2.7 de este documento.

4. IPoIM – Evolución del prototipo.

4.1. Concepto

El concepto detrás de IPoIM son los túneles IP. La idea esencial de IPoIM es precisamente establecer un túnel IP utilizando como transporte a la mensajería instantánea. Fue por ello que en el año 2005 se desarrolló el primer prototipo capaz de comunicar dos computadoras en redes distintas, a través de un túnel IP que usaba como transporte los servicios de mensajería instantánea conocidos (públicos como MSN Messenger[25] o Google Talk[22], o sistemas personales como Jabber[12])

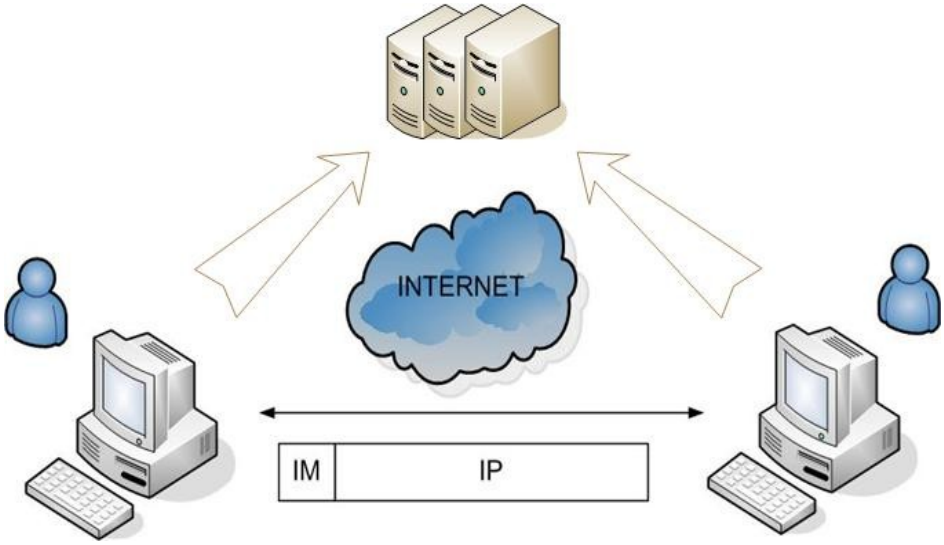

Figura 7: Comunicación establecida en IPoIM

Básicamente lo que hace el prototipo es enviar paquetes de red (binarios) codificados como texto plano, como si fuesen mensajes instantáneos. En la figura 7 se muestran dos instancias del prototipo comunicándose a través de Internet mediante algún servidor público de mensajería. Virtualmente el túnel se establece entre estas dos instancias, aunque el tráfico en realidad pasa a través del servidor público.

4.2. Diseño

IPoIM se diseñó utilizando dos productos free/open source existentes. Para resolver los requerimientos de envío de mensajes instantáneos se utilizó Gaim(Pidgin)[10], y para resolver los aspectos relacionados con los túneles se utilizó OpenVPN[9].

El diseño se basó en la integración de estas herramientas de manera de lograr un prototipo funcional que demostró que es posible establecer túneles IP a través de mensajería instantánea. La figura 8 muestra dicha integración.

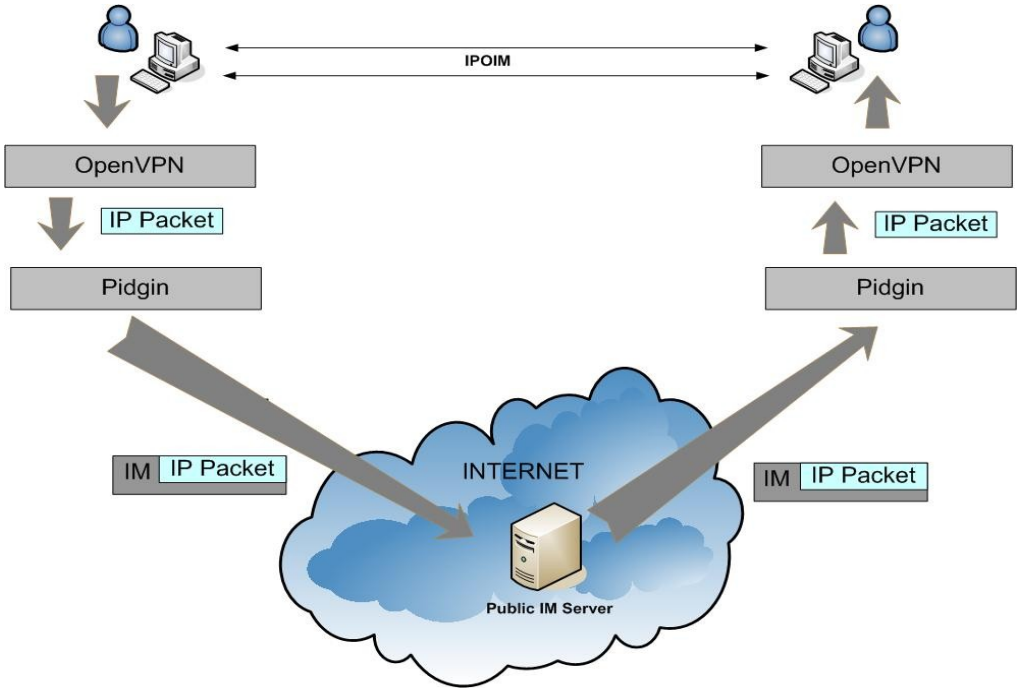

Figura 8: Integración de herramientas

En esta etapa nueva se determinó la necesidad de actualizar el prototipo con las versiones más recientes de ambas aplicaciones. En el caso de Gaim, el producto además de haberse actualizado, también sufrió un cambio de nombre ya que pasó a denominarse Pidgin. Actualmente Pidgin[10] se encuentra en la version 2.3.0. OpenVPN por su parte también se actualizó y la versión más reciente es la 2.0.9.

Threads

Otra decisión de diseño tomada para el desarrollo de IPoIM fue la de utilizar threads. Motivó esta decisión la necesidad de contar con un mecanismo que permita resolver el envío y la recepción de mensajes en forma simultánea y asincrónica. Esto es, se necesitaba tener un proceso que esté recepcionando los mensajes de la conversación de mensajería, para luego pasarlos a la capa de aplicación (en el modelo de referencia OSI), y otro proceso que espere a que existan datos listos para ser transmitidos por la red, de manera de interceptarlos para enviarlos como mensajes instantáneos (previa codificación a texto).

El problema que surge a partir de esta necesidad es que, como IPoIM se crea a partir de dos aplicaciones, es necesario hacer interactuar los mecanismos de ambas aplicaciones. Por un lado Pidgin utiliza un modelo de eventos provisto por GTK+[8]. Por este motivo cada acción que se toma es consecuencia de un evento generado por un suceso que ocurrió. Por ejemplo, el ingreso de texto del usuario en la ventana de la conversación genera un evento que es manejado por una función, la llegada de un mensaje instantáneo genera un evento que también debe ser

manejado por una función específica, y así con cada suceso que ocurre. Por otro lado OpenVPN no utiliza un modelo de eventos. En su lugar resuelve el problema con esperas en bucles del programa y con intervalos de tiempo. Con esta metodología OpenVPN espera en un bucle, que se le informe de que han llegado datos por el canal de comunicación, o que existen datos listos para trasmitir por el canal.

Debido a estas dos metodologías diferentes fue que se pensó en utilizar threads diferentes de ejecución. Un thread del programa se encarga de "escuchar" los eventos que ocurran, principalmente el evento que mas nos interesa es el que indica la llegada de un mensaje instantáneo. El otro thread de ejecución es el que realiza la espera en el bucle del programa esperando a que hayan datos nuevos para transmitir, de manera de poder enviarlos como mensajes instantáneos.

4.3. Mejoras del prototipo identificadas en el proyecto anterior.

El prototipo fue construido con el objetivo de ser funcional. Es por esta razón que para convertirlo en un producto se deben tener en cuenta algunas de las omisiones realizadas durante su implementación inicial:

• Restringir los envíos y recepciones solamente a los usuarios del túnel. En las implementaciones sobre la mensajería, no se verifica en ningún momento que el usuario que envía un mensaje sea el usuario con el cual se estableció el túnel.

• Avanzar en la adaptación del Gaim, a efectos de aislar los recursos utilizados del mismo. Por ejemplo, se podrían remover todas las implementaciones para las interfaces gráficas.

• Ídem para OpenVPN. El ejemplo en este caso podría ser remover todas las implementaciones sobre conexiones TCP o UDP reales.

• Corregir los errores no sistemáticos detectados en las pruebas. Estos errores no determinísticos implicaban la caída de la aplicación, y por lo tanto del túnel. No se pudo detectar en el proyecto anterior la causa de estos errores. Por ello una mejora al prototipo sería identificar las causas de estos errores y corregirlos.

• En la versión de plataforma Linux, el prototipo obliga a tener el servidor X levantado, y además tener permisos para desplegar cosas en él. Para mejorarlo se deberían eliminar estas dependencias, y permitir que funcione como demonio o desde consola de texto.

• Corregir la finalización de la aplicación, para que ésta pueda ser terminada mediante comandos.

• Implementar algún sistema de reintento ante caídas en la conexión con la mensajería instantánea.

• La versión en la plataforma Windows, obligó al prototipo a ser una librería para poder utilizar los protocolos de mensajería instantánea. Para mejorarlo se podrían modificar los protocolos o utilizarlos mediante otro mecanismo.

• La instalación del prototipo no es adecuada ya que utiliza archivos de instalación generados en una máquina en particular. Esto hace que existan incosistencias a la hora de ejecutar los scripts de instalación en otra maquina. Sería bueno utilizar las `autotools`[7]. Estas herramientas, presentes en cualquier sistema unix o clónico, permiten generar los archivos make, así como otros scripts de configuración en cada máquina que se desee instalar el prototipo.

4.4. Implementación

En esta sección se detallan los aspectos relacionados con la implementación del prototipo. Por tratarse de un prototipo desarrollado en un proyecto de grado anterior[35], se decidió que en esta sección se enumeren sólo las mejoras realizadas al prototipo existente, detallando si es necesario, las características implementadas en el proyecto anterior.

Migración de versión de las aplicaciones utilizadas (Gaim y OpenVPN)

El primer objetivo del presente proyecto de grado fue actualizar el prototipo con las versiones nuevas de los productos utilizados. Esto implicó una reimplementación de las partes de código que utilizaban funciones que fueron modificadas con las nuevas versiones de los productos.

Gaim (Pidgin)

Gaim (actualmente Pidgin) fue el producto que más cambios sufrió en el transcurso del tiempo entre que se desarrolló la primera versión del prototipo IPoIM (2005) y el momento actual. En primer lugar cambió su nombre. Pasó a llamarse Pidgin, y aunque parezca un detalle menor, este cambio significó un cambio de nombre en todas las funciones implementadas en el producto. Desde la librería libgaim, que pasó a denominarse libpurple, hasta cada una de las operaciones que tenían el prefijo `gaim_nombreoperacion`, que pasaron a ser `pidgin_nombreoperacion`. Estos cambios implicaron una revisión completa del código en busca de sentencias a modificar.

Para implementar IPoIM fue necesario realizar modificaciones a los archivos fuente del programa, de manera de permitir la integración de las herramientas para el correcto funcionamiento del túnel IP.

Los principales archivos a los que se le realizaron modificaciones, y una breve descripción de las mismas se presentan a continuación:

`conversation.c`	Se agrega un argumento a la función que registra el manejador del evento “received-im-

	msg”
`pounce.c`	Se comenta la línea que asocia una función que manejaba el evento “received-im-msg”. Ahora dicho evento es manejado por otra función.
`gtkmain.c`	Se comentó la función principal (main) y además se renombró a `ipoim-gtkmain.c`.

OpenVPN

OpenVPN en su última versión también sufrió modificaciones. Por lo que se debieron analizar los cambios realizados para verificar si afectaban o no el funcionamiento de IPoIM. Para ello se tomaron los archivos nuevos y se fueron incorporando las modificaciones realizadas en la antigua versión de OpenVPN, de manera de obtener el mismo resultado, un prototipo funcional de IPoIM.

Los principales archivos que sufrieron modificaciones son:

`error.c`	Se modifica la inclusión del archivo status.h ya que el mismo tiene por nombre ipoim-status.h
`forward.c`	Se modifica la inclusión de varios archivos. Se modifica la parte del código que realiza el envío de paquetes por la red a través de un socket. En lugar de socket se realiza el envío a través de la mensajería instantánea. Para realizar el envío se codifica y fragmenta el paquete en tantos mensajes como sea necesario.
`openvpn.c`	Se modifica la inclusión de varios archivos. Se escribe el código que genera los subprocesos que manejan el envío y recepción de los mensajes instantáneos. Se modifica el proceso principal (main) para que inicialize correctamente la aplicación Pidgin y luego se ejecuten los subprocesos antes mencionados.
`options.c`	Se modifican las opciones que el programa recibe en la línea de comandos de manera de que pueda recibir los parámetros de configuración de IPoIM.
`sig.c`	Se modifica la inclusión del archivo status.h ya que el mismo tiene por nombre ipoim-status.h. Se modifica el manejador de señales para que adminta la terminación del programa capturando la señal SIGINT.

forward.c y openvpn.c son los archivos que más modificaciones sufrieron. Es por ello que se anexan íntegramente al final del documento.

Y se agregaron los archivos

`dicc.c`	Contiene la implementación de un nuevo método de codificación. El método diccionario.
`encoding.c`	Este archivo contiene lo referente al manejo de métodos de codificación.
`gaim_ipoim.c`	Aquí se implementan las operaciones para el manejo de Pidgin. Inicialización, operaciones para el manejo de la aplicación de mensajería.

Estos archivos agregados se pueden consultar en los anexos a este documento, donde se presenta el código fuente de los mismos.

Otras consideraciones

Además de las modificaciones realizadas al código de ambas aplicaciones, fue necesario renombrar algúnos archivos que existían tanto en Pidgin así como en OpenVPN y llevaban el mismo nombre. Esto fue necesario ya que para utilizar las `autotools`[7] era imprescindible que no existieran archivos de código fuente con el mismo nombre para evitar ambigüedades.

La lista de archivos renombrados es la siguiente:

```
ntlm.c
ntlm.h
plugin.c
plugin.h
proxy.c
proxy.h
status.c
status.h
```

Estos archivos son parte de Pigdin, así como de OpenVPN. Por ello se decidió renombrar los archivos de OpenVPN de la siguiente manera:

```
ipoim-ntlm.c
ipoim-ntlm.h
ipoim-plugin.c
ipoim-plugin.h
ipoim-proxy.c
ipoim-proxy.h
ipoim-status.c
ipoim-status.h
```

Correcta inicialización de la librería libgcrypt.

Debido a que IPoIM utiliza threads para resolver el envío y la recepción de mensajes simultáneamente, se tuvieron que considerar los aspectos relacionados con el código, de manera de verificar el correcto funcionamiento en entornos multithreadeds.

Para el caso de la librería `libgcrypt`, cuando se utiliza en entornos multithreadeds, se debe prestar especial atención a la inicialización de la misma. Esto es debido a que, mas allá de que la librería fue diseñada siguiendo la metodología "thread safe", algunas partes de la misma no lo son, como es el caso del generador aleatorio utilizado en la parte criptográfica. Por este motivo es necesario registrar las funciones de callback para asegurar el correcto funcionamiento de la librería[6].

Existen macros que ayudan a la inicialización de la librería para los casos en que la aplicación multithread utiliza `Posix Threads`. También se disponen de macros para aplicaciones que utilizan `GNU PTH Threads`. Sin embargo en nuestro caso, IPoIM utiliza `Glib Threads` por lo que fue necesario escribir las funciones a registrar de la siguiente manera:

```
static int gcry_gthread_mutex_init(void **priv)
{
        GMutex *lock = g_mutex_new();
        if (!lock)
                return ENOMEM;
        *priv = lock;
        return 0;
}

static int gcry_gthread_mutex_destroy(void **lock)
{
        g_mutex_free(*lock);
        return 0;
}

static int gcry_gthread_mutex_lock(void **lock)
{
        g_mutex_lock(*lock);
        return 0;
}

static int gcry_gthread_mutex_unlock(void **lock)
{
        g_mutex_unlock(*lock);
        return 0;
```

```
}

static struct gcry_thread_cbs gcry_threads_gthread = {
       GCRY_THREAD_OPTION_USER, NULL,
       gcry_gthread_mutex_init, gcry_gthread_mutex_destroy,
       gcry_gthread_mutex_lock, gcry_gthread_mutex_unlock,
       NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL
};

gcry_control(GCRYCTL_SET_THREAD_CBS, &gcry_threads_gthread);
```

Esta última función es la que registra las funciones de callback definidas anteriormente, y permite la correcta inicialización de la librería.

Con esta inicialización se corrige uno de los errores detectados en la versión anterior del prototipo (proyecto de grado 2005) enumerados anteriormente, en el que se reconocía que la aplicación presentaba "errores no sistemáticos" que determinaban la caída del túnel de manera no determinística.

Eliminación de la dependencia del entorno gráfico.

IPoIM, por tratarse de un prototipo desarrollado en base a dos aplicaciones existentes, está sujeto a sus caraterísticas y limitaciones. Pidgin por su parte, es una aplicación con interfaz gráfica desarrollada con GTK (GIMP toolkit) por lo que utiliza muchas funcionalidades que para el caso de IPoIM no son necesarias. Por ejemplo el manejo de ventanas gráficas no es utilizado en IPoIM. En cambio, el manejo de eventos provisto por GTK sí es utilizado.

Es de interés para este proyecto eliminar la dependencia del entorno gráfico existente en el prototipo anterior.

Para lograr la independencia del entorno gráfico se buscó reescribir el código que inicializa la librería GTK+. La solución fue utilizar la función provista que realiza la inicialización del entorno gráfico, pero que en caso de no detectar un gestor de ventanas ejecutándose, no aborta la ejecución, si no que permite la continuación de la ejecución de la aplicación. Esto permite ejecutar el prototipo en un ambiente sin entorno gráfico, y al mismo tiempo, brinda el manejo de eventos que es indispensable para que la aplicación funcione correctamente.

La inicialización se realiza de la siguiente manera:

```
gtk_init_check(&argc, &argv);
```

Si no se detecta el entorno gráfico ejecutándose, de igual manera la aplicación continúa su ejecución, y con todas las posibilidades de manejo de eventos antes mencionadas.

Luego se invoca a la función

```
gtk_main();
```

Que es la encargada de "escuchar" los eventos ya que es el bucle principal de las aplicaciones GTK+.

Nuevo método de codificación. Diccionario de palabras.

Se planteó la necesidad de buscar alternativas de codificación para enviar información mediante la mensajería instantánea, y al mismo tiempo que dicha información sea encubierta, de manera que no pueda ser detectada por los filtros de contenido. Esta necesidad derivó en un método de codificación que intenta parecerse a una conversación entre seres humanos. Fue entonces que se decidió codificar utilizando palabras válidas que las personas utilizan para comunicarse. De esta manera los filtros de contenido no puden distinguir entre las palabras que una persona teclea en una conversación, y las palabras generadas por IPoIM para codificar bytes.

Con esta idea se creó el método de codificación con diccionario de palabras. El mismo es presentado con más detalles en el capítulo 5 de este documento.

Fragmentación de mensajes.

En el proyecto realizado en el año 2005, se determinó empíricamente cual era el máximo valor de MTU (maximum transfer unit) que permitía establecer el túnel entre las dos instancias de IPoIM de manera que la comunicación se establezca a través de la mensajería instantánea. El valor determinado en ese entonces fue de 1079 bytes por paquete. Este valor es especificado en la configuración del túnel.

Esta limitación del valor de MTU impedía utilizar métodos alternativos de codificación, ya que el tamaño de los mensajes varían según cual sea el método utilizado para codificar. Entonces el utilzar otro método, (y teniendo en cuenta que el mtu se adapta al medio por el cual se realiza la comunicación), implica tener que limitar el tamaño del paquete al tamaño de MTU utilizado. Para el caso de ethernet, el MTU es de 1500 bytes, mientras que para el caso de PPPoE (Point to Point over Ethernet) el MTU es de 1492 bytes.

El método de codificación con diccionario de palabras descripto anteriormente genera mensajes bastante mas largos que los mensajes habituales. Por ese motivo fue necesario implementar un método de fragmentación de mensajes capáz de manejar los mensajes generados por este método de codificación.

El método de fragmentación implementado particiona los mensajes según un largo determinado que se pasa por parámetro. La idea de este método es dividir los

mensajes largos en varios mensajes con un tamaño adecuado para ser enviado en un medio de comunicación como puede ser Ethernet o PPPoE.

El esquema de fragmentación que se utilizó es el siguiente:

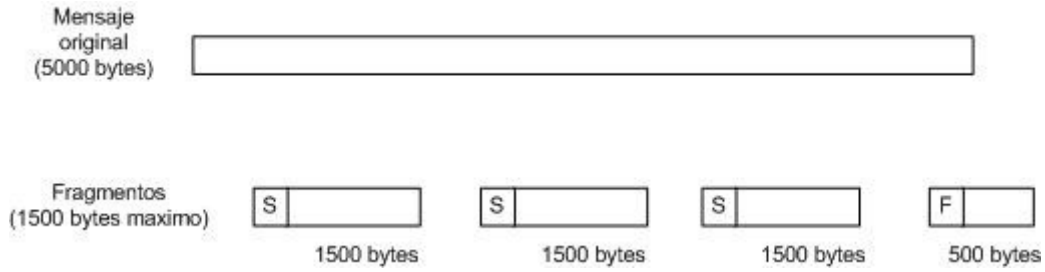

Figura 9: Fragmentación de mensajes

Se definieron dos constantes que determinan el largo y la cantidad de los fragmentos, y una tercera constante que indica un desplazamiento donde se coloca una "marca" a los fragmentos que indica si existen más fragmentos o se trata del último fragmento de un mensaje.

En el ejemplo de la figura 9, los parámetros serían:

Largo: 1500 bytes (largo máximo permitido)

Cantidad de fragmentos: 10 (cantidad máxima permitida, sólo se generan 4 fragmentos en el ejemplo)

Desplazamiento: 1 (se utiliza 1 byte para determinar si se trata del útimo fragmento de un mensaje o en caso contrario, existen más fragmentos. En la figura la "F" indica que es el último fragmento, mientras que la "S" significa que siguen mas fragmentos)

Las funciones implementadas para resolver la fragmentación son las siguientes:

```
/* Esta funcion recibe un mensaje y lo particiona en mensajes de largo
maximo "size". El resultado se carga en la matriz "msgs" */
int spwn(char* msg, int size, char** msgs){
      int i=0;
      int j=OFFSET;
      int k=0;
      char code_sigue=CODE_SIGUE;
      char code_end=CODE_END;

      while(k<size){
            while(j<MSG_LARGO+OFFSET && k<size){
                  msgs[i][j]=msg[k];
                  j++;
                  k++;
```

```
            }
            msgs[i][j]='\0';

            if (k<size){
                  msgs[i][0]=code_sigue;
                  }
            else{
                  msgs[i][0]=code_end;
                  }
            i++;
            j=OFFSET;
      }
      int c=i;

      while(i<MSG_CANT){
            msgs[i]=NULL;
            i++;
            }
      return c;
}

/*Esta funcion recibe un arreglo con todos los fragmentos de un
mensaje y reconstruye el mismo, cargandolo en la variable "msg" */
int unspwn(char** msgs, char* msg){
      int i=0;
      int cnt=(int)*msgs[0];
      msgs[0]++;
    while (i<cnt){
          strcat(msg,msgs[i]);
          i++;
    }
}
```

Mutuoexclusión del canal de comunicación.

Un aspecto a corregir del prototipo anterior, fue que no se verificaba que los mensajes recibidos por mensajería instantánea fuesen mensajes pertenecientes a la comunicación establecida en el túnel IP. En consecuencia, cualquier mensaje recibido fuera del túnel causaba la caída del mismo, ya que introducía ruido en la comunicación.

Para resolver este inconveniente se buscó la forma de validar los mensajes recibidos con los ususarios involucrados en el túnel. Es decir, sólo se aceptan los mensajes enviados por el usuario de mensajería con el que se estableció el túnel. El resto de los mensajes recibidos son descartados.

Esto se resolvió "marcando" los mensajes que son enviados por el túnel de manera que la contraparte pueda determinar que el mensaje es un mensaje válido de la comunicación, o se trata de un mensaje ajeno a la misma.

La marca de los mensajes consiste en un código que se agrega al principio de cada mensaje. Éste código cumple dos funciones.

1- Distingue al mensaje del resto de los mensajes instantáneos, ya que un mensaje instatáneo común, no tiene este código al principio.

2- A su vez el mismo código es utilizado para la fragmentación de mensajes ya que un mensaje fragmentado, se envía en varias partes las cuales se deben distinguir. El código utilizado distingue entre dos opciones. O se trata del último fragmento del mensaje, o es un fragmento intermedio.

Cómo el mensaje que se envía es texto, y cómo sólo se necesita distinguir entre dos valores, se decidió codificar utilizando un número que corresponde a un carácter no desplegable del código ASCII. De esta manera, se logra enviar el código del mensaje en el mismo mensaje, y se obtiene de esta manera una forma de distinguir los mensajes que llegan a través de la mensajería instantánea.

Modificación de la forma en que se especifica el método de codificación.

En el proyecto anterior se especificaba el método de codificación (único hasta ese momento) de manera estática dentro del código fuente. Esto no era práctico ya que para cambiar el método de codificación era necesario recompilar el programa.

Sin embargo, se implementó pensando en la posibilidad de extender los métodos de codificación con otras alternativas. Por ello se implementaron operaciones de codificación y decodificación que reciben como parámetro el tipo de codificación a utilizar.

Luego con los desarrollos realizados en el actual proyecto se hizo necesario especificar dicho parámetro en tiempo de ejecución, con lo cual se modificó el codigo fuente para que admita un parámetro más en el archivo de configuración de la aplicación.

Lo que se agregó fue un parámetro más a la línea del archivo de configuración donde se especifican los usuarios de mensajería y el protocolo a utilizar en el túnel.

El caso genérico sería:

```
ipoim usuario1 protocolo usuario2 metodoDeCodificacion
```

Por ejemplo:

```
ipoim asy-ipoim@fing.edu.uy prpl-msn mlr-ipoim@fing.edu.uy Base64
```

Más detalles de esta línea y de otras correspondientes al archivo de configuración pueden encontrarse en el apéndice 3.

Finalización del programa.

Otra mejora realizada a la aplicación anterior fue en la finalización de la ejecución del programa. El prototipo anterior carecía de una finalización correcta y debía ser finalizado en forma abrupta utilizando algún mecanismo provisto por el sistema operativo.

Una finalización correcta debería liberar los recursos del sistema operativo (memoria, procesos, semáforos, etc), cerrar conexiones y devolver el control al sistema sin necesidad de tener que recurrir a métodos de finalización como pkill, kill, etc, para el caso de Linux por ejemplo.

Para esta nueva versión se realizó el manejo de las señales necesario para finalizar la aplicación en forma correcta, al recibir la señal SIGINT cuando el usuario tipea la secuencia CTRL-C. Se registraron manejadores de las señales capaces de capturar la misma y realizar la finalización correcta del programa.

4.5. Instalación

Otra mejora realizada al prototipo IPoIM realizado en el proyecto de grado anterior fue el instalador. Se logró un instalador muy sencillo de utilizar, al estilo de los instaladores de aplicaciones para Linux. Se trabajó con las `autotools`[7] de manera de poder compilar, así como instalar el prototipo como cualquier otra aplicación de Linux. Más información acerca de la instalación del producto se puede encontrar en el apéndice 3.

5. Estudio e implementación de diferentes algoritmos de codificación.

IPoIM es una aplicación capaz de establecer túneles IP a través de las redes de mensajería públicas. Como ya fue presentado, esto lo logra embebiendo paquetes binarios (paquetes IP) dentro de los mensajes instantáneos de manera de hacerlos viajar a través de redes públicas como lo son MSN Messenger, o Google Talk, o incluso a través de servidores privados de mensajería implementados con Jabber.

Para lograr este cometido, fue necesaria una conversión en los datos que se querían enviar de manera que puedan viajar como "mensajes instantáneos". Fue entonces que se determinó que los paquetes no podrían ser enviados como un mensaje si no se convertían previamente a mensajes de texto puro. Por consiguiente se resolvió enviar los mensajes codificados de forma tal que los paquetes de red se transformen en mensajes de texto puro.

En el proyecto de grado realizado en el año 2005 se utilizó Base64 como codificación para enviar datos a través de la mensajería instantánea. Esto fue motivado por el hecho de que era indispensable enviar datos utilizando caracteres desplegables, los cuales pueden ser enviados por mensajería instantánea. La opción utilizada entonces fue la codificación en Base64 ya que la misma está disponible y es ampliamente utilizada por varias aplicaciones como el correo electrónico, etc.

Un objetivo de este proyecto es investigar las posibles alternativas de codificación que se pueden utilizar para enviar datos utilizando como transporte a la mensajería instantánea.

El presente capítulo del documento abarca los aspectos relacionados a las técnicas de codificación utilizadas por IPoIM para enviar información binaria a través de la mensajería instantánea. En la sección 5.1 se detallan las técnicas utilizadas para codificar y decodificar, presentando descripciones detalladas y ejemplos.

También se analizan en la sección 5.2 las técnicas de detección y/o filtrado de mensajería para cada método de codificación. Con estas técnicas se intenta detectar el tráfico generado por IPoIM y eventualmente filtrarlo.

Luego en la sección 5.3 se presentan una serie de pruebas realizadas con herramientas existentes de filtrado, verificando si se pudo detener el tráfico de IPoIM o si éste no se detectó.

5.1. Métodos de codificación.

El éxito de IPoIM depende de la capacidad de éste de poder ser trasportado por los sistemas de mensajería instantánea, y no ser detectado y/o filtrado. Para no ser detectado es necesario utilizar técnicas de codificación más refinadas, de manera de pasar inadvertido por los sistemas de gestión de mensajería. En el proyecto de grado realizado en el año 2005 se utilizó base64 como método de codificación. Si bien este método fue efectivo para demostrar la viabilidad de la solución, también es detectable fácilmente.

Teniendo en cuenta que en este proyecto apuntamos a la seguridad, es de interés demostrar distintas alternativas de codificación. De esta manera se puede analizar si es posible detectar o no, el tráfico generado por IPoIM, utilizando las herramientas existentes desarrolladas para brindar seguridad a las aplicaciones de mensajería instantánea.

Para ello se analizan las técnicas existentes de codificación de código binario a texto plano. Al mismo tiempo se detallan las técnicas utilizadas en este proyecto, explicando como se implementan, y dando ejemplos de las mismas.

5.1.1. Técnicas de codificación de binario a texto.

Hexadecimal

Hexadecimal es un sistema de numeración con base 16 utilizado para representar código binario utilizando un conjunto de caracteres reducido, de manera de facilitar la lectura para los seres humanos. Se puede considerar como un método de codificación muy simple que codifica un byte en dos caracteres del conjunto (1,2,3,4,5,6,7,8,9,A,B,C,D,E,F).

Uuencode

Uuencode es una forma de codificación de binario a texto originada en el sistema operativo Unix, utilizada para transmisión de datos sobre el sistema de correo electrónico UUCP. Uuencode se volvió popular debido a que fue ampliamente usado para el envío de datos binarios en el correo electrónico. Esta técnica fue reemplazada por MIME y yEnc.

MIME

Multipurpose Internet Mail Extensions (MIME) es un estándar que extiende las capacidades del correo eléctronico para incorporar otros conjuntos de caracteres distintos de ASCII, archivos adjuntos binarios, mensajes multi parte e información en los headers que no esté en ASCII. Entre las posibilidades de MIME justamente se encuentra la capacidad de codificar datos binarios para que puedan ser enviados utilizando el conjunto de caracteres de 7-bits ASCII. MIME es un formato que define

ciertos parámetros, no es un método de codificación en si mismo. Para codificar utiliza técnicas como "quoted-printable" o "base64".

yEnc

yEnc es otro método de codificación de código binario a texto que utiliza un conjunto de caractedes extendido de 8 bits para codificar. El resultado de esto es que no tiene el overhead que tienen los métodos como base64 (33%) ya que codifica un byte con un carácter, que equivale a un solo byte. Este hecho resulta en mensajes codificados mas pequeños, con lo cual los mensajes pueden ser enviados más rápidamente. Sin embargo esta técnica adolece de problemas cuando los mensajes son enviados a través de sistemas que sólo pueden manejar conjuntos de caracteres de 7-bits, lo cual provoca corrupción de datos. Por otro lado no funciona muy bien con el popular conjunto de caracteres UTF-8.

BinHex

BinHex (Binary to Hexadecimal) es un método de codificación de binario a hexadecimal utilizado por el sistema operativo MacOS para enviar archivos binarios a través del correo electrónico.

UTF-8

UTF-8 (8 bit – Unicode Transformation Format) es una técnica de codificación de binario a texto de largo variable para Unicode. Utiliza de 1 a 4 bytes por símbolo para representar un alfabeto universal. UTF-8 se utiliza mucho para el envío de datos binarios a través del correo electrónico.

Base64

El esquema de codificación Base64 permite convertir datos binarios a una representación basada sólo en ASCII. Se usa en MIME para enviar ficheros por correo electrónico, entre otros usos. Se utiliza 64 como base ya que es la mayor potencia de 2 que permite utilizar caracteres ASCII desplegables.

Radix-64

Radix-64 es también conocido como "ASCII Armor" y es idéntico a la codificación de base64 descripta en MIME, con la única diferencia que agrega una suma de verificación de 24 bits. Esta suma de verificación es calculada con los datos de entrada y luego codificada con el mismo algoritmo base64 y agregada al final de los datos codificados, separada por un carácter "=".

5.1.2. Esquemas de codificación utilizados por IPoIM

Las técnicas de codificación antes mencionadas en general son similares en cuanto a que codifican bytes con caracteres desplegables, de manera de poder ser enviados a través de medios como el correo electrónico. A los efectos de este proyecto, estas técnicas pueden agruparse ya que todas pueden ser detectadas de

la misma manera, es decir, utilizando patrones. Por ello no se implementaron éstas técnicas, debido a que ya se cuenta con una codificación de este tipo (Base64) con la cual se pueden realizar las pruebas de filtrado. En lugar de probar con otras técnicas conocidas (además de base64) se decidió utilizar una nueva técnica que codifique los datos binarios con palabras reales de un idioma existente. De esta manera se busca evadir los filtros de contenido que poseen las herramientas de gestión de mensajería instantánea. Esta nueva técnica se denominó "Diccionario" y se detalla a continuación, junto con el detalle de la técnica "Base64" que es también utilizada.

5.1.2.1. Base64

Funcionamiento.

Codificación

- Se toman 3 bytes de la entrada.
- Se juntan esos 3 bytes y los 24 bits resultantes se separan en cuatro grupos de 6 bits.
- Cada uno de esos grupos representa un número entre 0 y 63, que se usa como índice en la siguiente tabla para ser sustituido por un carácter: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 +/
- Esos cuatro caracteres se vuelcan en la salida.
- Se repite el proceso anterior con cada bloque de tres bytes hasta llegar al último bloque, si fuese menor de tres bytes entonces se rellenaría con ceros procediendo como antes. Por último si los últimos cuatro caracteres obtenidos son C0, C1, C2 y C3 se añade:
  - si el último bloque tenía 1 byte: C0, C1, =, =
  - si el último bloque tenía 2 bytes: C0, C1, C2, =
  - si el último bloque tenía 3 bytes: C0, C1, C2, C3.

  Normalmente cada 76 caracteres de la salida se pone un salto de línea aunque esto no es parte de la definición de Base 64.

Decodificación

Lógicamente el proceso de decodificación es inverso al anterior, cada cuatro caracteres producirán 3 bytes, con la precaución de interpretar los últimos cuatro como queda dicho arriba.

Ejemplo #1 – Codificación de bytes usando Base64.

Si se quiere codificar la siguiente secuencia de bytes:

0xef 0xba 0xbc 0x95 0x5f

Primero se agrupan de tres en tres rellenando el último bloque con ceros:

0xefbabc
0x955f00

Ahora escribimos cada bloque en binario (24 bits)

111011111011101010111100
100101010101111100000000

Que ahora dividimos en bloques de 6 bits

111011 111011 101010 111100
100101 010101 111100 000000

Que en decimal son

59 59 42 60
37 21 60 0

y corresponden a los caracteres

7 7 q 8
l V 8 A

Y como el último bloque es de 2 bytes la cadena de salida sería

77q8lV8=

respetando sólo los tres últimos caracteres y rellenando con "="

Pruebas de performance.

A continuación se presentan una serie de pruebas realizadas con la finalidad de medir la performance del método de codificación Base64. Para realizar las pruebas se utilizó la herramienta NetPerf[5].

Las pruebas que se realizaron son:

- TCP_STREAM – Obtiene datos de performance (throughput) a partir de datos predefinidos por la aplicación.
- TCP_SENDFILE – Realiza la misma prueba que TCP_STREAM pero en este caso se utiliza un archivo de datos pasado como parámetro. (En los ejemplos el archivo utilizado tenía un tamaño de 1,12 MBytes)
- TCP_RR – Este test realiza pruebas de Request/Response.
- TCP_CC – Mide la velocidad de inicio y cierre de conexiones TCP.

Test Nro1.

netperf -t TCP_STREAM -H 192.168.0.1 -f k -l 60

Socket Size (bytes)	Socket Size (bytes)	Message Size (bytes)	Elapsed Time (secs)	Throughput (10^3bits/sec)
87380	16384	16384	92.92	31.01
87380	16384	16384	85.88	35.82
87380	16384	16384	82.30	37.91

Los resultados de esta prueba indican que en promedio se alcanza un rendimiento de 34,91 x10^3 bits por segundo, o lo que es lo mismo 34,9 Kbits/seg

Test Nro2.

netperf -t TCP_SENDFILE -F netperf_2.4.4.tar.gz -H 192.168.0.1 -f k -l 60

Recv Socket Size (bytes)	Send Socket Size (bytes)	Send Message Size (bytes)	Elapsed Time (secs)	Throughput (10^3bits/sec)
87380	16384	16384	92.07	30.47
87380	16384	16384	79.22	33.26
87380	16384	16384	90.02	34.41

En este caso, y utilizando un archivo como fuente de datos se alcanzó un promedio de 32,71 x10^3 bits por segundo, es decir, 32,7 Kbits/seg

Test Nro3.

netperf -t TCP_RR -H 192.168.0.1 -f k -l 60

Socket Send	Size Recv	Request Size	Resp. Size	Elapsed Time	Trans. Rate

(bytes)	(Bytes)	(bytes)	(bytes)	(secs)	(per sec)
16384	87380	1	1	60.01	0.03
16384	87380	1	1	60.01	0.03
16384	87380	1	1	60.01	0.03

Para los tests de request/response los resultados obtenidos fueron de 0,03 transacciones por segundo.

Test Nro4.

netperf -t TCP_CC -H 192.168.0.1 -f k -l 60

Socket Send (bytes)	Size Recv (Bytes)	Request Size (bytes)	Resp. Size (bytes)	Elapsed Time (secs)	Trans. Rate (per sec)
16384	87380	1	1	60.01	0.62
16384	87380	1	1	60.01	0.50
16384	87380	1	1	60.01	0.52

En este caso las transacciones por segundo fueron de 0,54 t/seg promedialmente.

5.1.2.2. Diccionario de palabras.

El método de codificación con diccionario de palabras consiste en la codificación de un flujo de bytes mediante palabras válidas de un idioma. Para probar dicho método se utilizaron palabras del idioma inglés. Más precisamente se codifica cada byte con una palabra. No es un método eficiente ya que presenta un incremento considerable en el flujo de información necesario para enviar unos bytes por el canal de comunicación. De todas maneras el objetivo que se planteó fue evadir controles aplicados por filtros de contenido.

Funcionamiento.

El esquema de codificación es el siguiente:

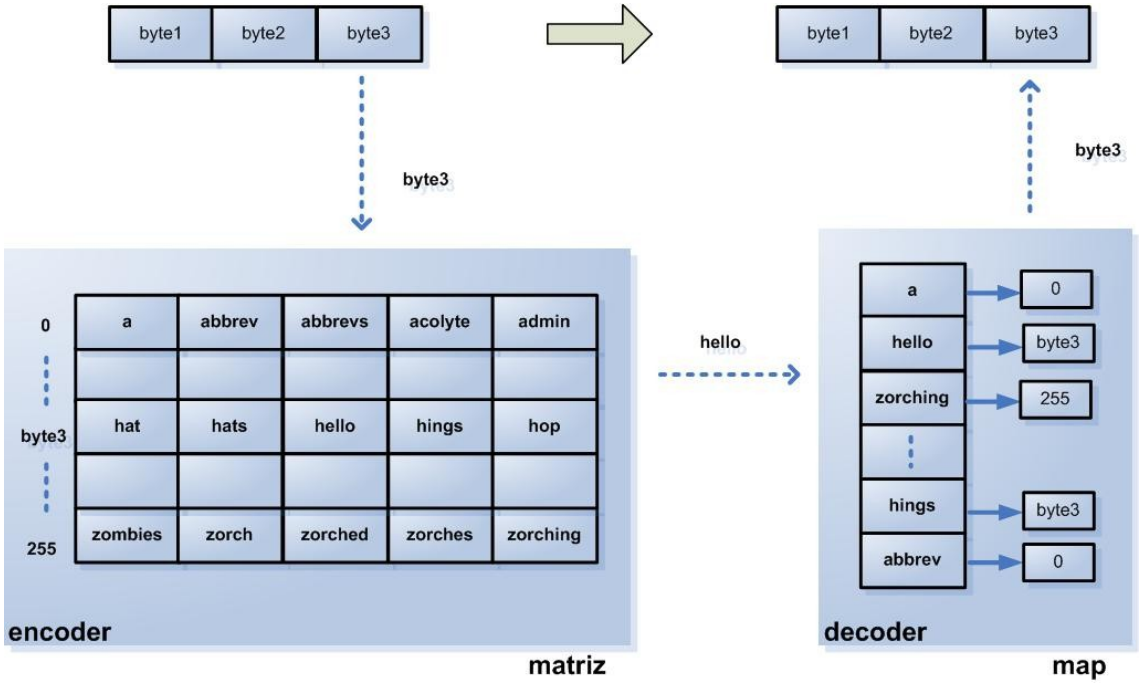

Figura 10: Esquema de codificación con Diccionario de palabras

Por un lado se tiene un diccionario de palabras. Este diccionario se carga en una matriz de cadenas de caracteres de dimensión 256x5. La dimensión de la matriz se debe a que para codificar cada byte (de 256 posibles) se deben alojar 5 palabras, de manera de poder utilizar las mismas. A su vez, por cada palabra en la matriz de strings se da de alta en un map el valor de la fila correspondiente a dicha palabra. La matriz es usada por el método de codificación y el map por el método de decodificación.

Como se mencionaba anteriormente, ste algoritmo de codificación utiliza 5 palabras para representar cada byte, por lo que admite varias codificaciones para un mismo byte. Por ejemplo si se tienen 5 bytes para codificar existen 5^5 posibles codificaciones. Esto asegura que no se pueda detectar este tipo de codificación utilizando un patrón, ya que no se codifica de igual manera un mismo byte en distintas codificaciones.

En la figura 10 se muestra el funcionamiento cuando se transmite el tercer byte de una serie de bytes, representado en la figura con el nombre "byte3". El byte3 se podría codificar como cualquiera de las palabras en la fila <byte3> de la matriz, es decir (hat, hats, hello, hings o hops). Para elegir la palabra se obtiene un número al azar entre 0 y 4 que determina la columna elegida y consecuentemente, la palabra ubicada en la columna sorteada. En este caso el numero de columna elegido al azar fue el 2 por lo que la palabra que codifica al byte3 es "hello". Luego en la decodificación se obtiene el valor del byte3 cuando se busca en el map la palabra "hello".

Codificación:

Para cada byte que se desea transmitir se toma el valor decimal de dicho byte(de 0 a 255) y con él se obtiene el número de fila de la matriz diccionario. Luego se obtiene un número aleatorio entre 0 y 4 para determinar la columna de la matriz y con ello la palabra que ocupa en esa posición (fila, columna). Luego se envía dicha palabra por el canal.

Decodificación:

En el otro extremo, para decodificar el byte encubierto en la palabra, se busca dicha palabra en el map, y se obtiene el valor del byte asociado (0 a 255).

Ejemplo #2 – Codificación de bytes usando Diccionario.

Dada la misma secuencia de bytes del ejemplo #1, si se la quiere codificar utilizando el método diccionario de palabras se haria de la siguiente manera:

0xef 0xba 0xbc 0x95 0x5f

En binario seria

11101111 10111010 10111100 10010101 01011111

Que en decimal son

239 186 188 149 95

y si tenemos el siguiente diccionario en una matriz (no se muestra completo):

	0	1	2	3	4
0	a	abbrev	abbrevs	accumulator	accumulators
1	acolyte	acolytes	admin	admins	alt
2	alts	amoeba	amoebae	amoebas	and
....	....	....	....	....	....
....	....	....	....	....	....
95	foregrounded	foregrounding	foregrounds	forked	forum
....	....	....	....	....	....
....	....	....	....	....	....
149	modes	mods	modulo	monstrosities	monstrosity
....	....	....	....	....	....
....	....	....	....	....	....
186	rococo	rogue	rogues	root	roots
187	rude	ruder	rudest	runes	runic
188	sacred	saga	sagas	said	Salt
....	....	....	....	....	....
....	....	....	....	....	....
239	vanilla	virtual	virus	viruses	visionaries
....	....	....	....	....	....
....	....	....	....	....	....
255	zombies	zorch	zorched	zorches	zorching

Las filas que determinan los valores decimales de los bytes son las filas 239 186 188 149 y 95.

Por lo que los bytes se codificarían como:

matriz(239, rand(0..4)) = virtual
matriz(186, rand(0..4)) = roots
matriz(188, rand(0..4)) = salt
matriz(149, rand(0..4)) = modes
matriz(95, rand(0..4)) = foregrounded

Es decir una posible codificación serían las siguientes palabras :

virtual roots salt modes foregrounded

Pruebas de performance.

Del mismo modo que con el método de codificación Base64, para el método con Diccionario se realizaron los mismos tests, los cuales arrojaron los siguientes resultados:

Test Nro1.

netperf -t TCP_STREAM -H 192.168.0.1 -f k -l 60

Socket Size (bytes)	Socket Size (bytes)	Message Size (bytes)	Elapsed Time (secs)	Throughput (10^3bits/sec)
87380	16384	16384	94.33	7.00
87380	16384	16384	116.86	6.53
87380	16384	16384	105.05	7.11

En este caso se puede apreciar notoriamente que el método "Diccionario" es mucho menos eficiente que el de Base64 ya que se alcanzan en promedio unos 6,88 Kbits/seg.

Test Nro2.

netperf -t TCP_SENDFILE -F netperf_2.4.4.tar.gz -H 192.168.0.1 -f k -l 60

Recv Socket Size (bytes)	Send Socket Size (bytes)	Send Message Size (bytes)	Elapsed Time (secs)	Throughput (10^3bits/sec)
87380	16384	16384	117.37	6.37
87380	16384	16384	115.59	6.47
87380	16384	16384	106.14	7.04

Este test reafirma los resultados obtenidos en el test anterior. Una vez mas, el promedio de throughput fue 6,63 Kbits/seg. Esto demuestra que el método de codificación Diccionario no es eficiente en el uso del enlace, ya que con el método Base64 se alcanzaron throughputs cinco veces mayores en las mismas condiciones, es decir unos 30 Kbits/seg.

Test Nro3.

netperf -t TCP_RR -H 192.168.0.1 -f k -l 60

Socket Send (bytes)	Size Recv (bytes)	Request Size (bytes)	Resp. Size (bytes)	Elapsed Time (secs)	Trans. Rate (per sec)

16384	87380	1	1	60.01	0.02
16384	87380	1	1	60.01	0.02
16384	87380	1	1	60.01	0.02

En los test de request/response los resultados obtenidos son 0,02 transacciones por segundo.

Test Nro4.

netperf -t TCP_CC -H 192.168.0.1 -f k -l 60

Socket Send (bytes)	Size Recv (bytes)	Request Size (bytes)	Resp. Size (bytes)	Elapsed Time (secs)	Trans. Rate (per sec)
16384	87380	1	1	60.00	0.42
16384	87380	1	1	60.00	0.43
16384	87380	1	1	60.01	0.43

Mientras que en los tests de connect/close las transacciones por segundo fueron 0,43 t/seg.

5.2. Técnicas de detección y filtrado.

En la sección anterior se analizaron las técnicas de codificación utilizadas para enviar datos utilizando mensajería instantánea. Ahora es de interés analizar las técnicas para la detección y filtrado de ese tipo de tráfico.

Nuevamente, es de especial interés para este proyecto poder identificar cuando se están transmitiendo datos por un canal pensado para conversar entre personas.

Para desarrollar una técnica de filtrado se utilizó la herramienta IMSpector[26] presentada en el capítulo anterior, mientras que se utilizaron Symantec IM Manager[28] e IM Security Net[27] para detectar el tráfico de IPoIM.

5.2.1. Base64.

Para detectar si se trata de una codificación utilizando base64 se utilizó una heurística que sugiere que si una cadena de caracteres esta formada solamente por caracteres utilizados en codificaciones base64 en sus primeros N caracteres, entonces posiblemente se trate de una codificación en base64. Esta decisión se toma en base a que, lo que se debería estar transmitiendo por el canal debería ser una conversación entre personas. Por ello necesariamente deben aparecer caracteres fuera de la codificación base64, como el carácter " " (espacio en blanco)

entre otros. La elección del numero N se realiza de manera de considerar que no pueden haber palabras con largo mayor que N, con lo cual, si en los primeros N caracteres no hay por lo menos un espacio en blanco, muy posiblemente no se trate de una conversación, sino que podría ser un canal encubierto, utilizando el conjunto de caracteres que podrìa ser codificación base64.

Heurística de detección de codificación base64.

La heurística utilizada es la que se presenta a continuación.

```
/*
 * Los primeros n bytes del buffer buff son solo caracteres de
 * la codificacion base64? */
int is_b64_chars(const unsigned char *buf, size_t n) {
    const unsigned char *p;

    while (len > 0 && buf[len - 1] == '=')
        --len;
    for (p = buf+1; p < buf + len; ++p)
        if (!((*p >= 'A' && *p <= 'Z')
                || (*p >= 'a' && *p <= 'z')
                || (*p >= '0' && *p <= '9')
                || *p == '+' || *p == '/'))
            return 0;
    return 1;
}
```

La misma es aplicada a cada mensaje enviado.

Implementación.

La detección y filtrado de la codificación base64 se implementó como un plugin de la aplicación IMSpector, ya que la misma provee este mecanismo para adicionarle funcionalidades. Para ello se agregó el modulo base64contentplugin.cpp en el cual se encuentra implementada la heurística antes mencionada y la función contentfilter detallada a continuación:

```
/* The main plugin function. See contentplugin.cpp. */
int contentfilter(char *originalbuffer, char *modifiedbuffer)
{
      debugprint(localdebugmode, PLUGIN_SHORT_NAME ": filtering
before: original: %s",originalbuffer);
      if (is_b64_chars((const unsigned char *)originalbuffer,30)){
            char * s = modifiedbuffer;
            strcpy(modifiedbuffer,"Se detecto codificacion Base64");
            s=s+30;
            memset(s,(int) '*',strlen(originalbuffer)-30);
            }
      else strcpy(modifiedbuffer, originalbuffer);

      debugprint(localdebugmode, PLUGIN_SHORT_NAME ": filtering after:
modified: %s", modifiedbuffer);
```

```
        return 0;
}
```

Para aplicar el filtro de contenido se agregó en el archivo de configuración de IMSpector el parámetro:

```
# base64 filtering
enable_base64=yes
```

Dicho parámetro especifica que se desea aplicar el filtro de contenido. Si no se desea que la aplicación aplique dicho filtro, la opción en el archivo de configuración debe estar comentada.

```
# base64 filtering
#enable_base64=yes
```

La implementación completa del filtro se encuentra en el apéndice 1.

5.2.2. Diccionario de palabras.

Estadísticas de mensajes.

La técnica de detección mediante estadísticas de mensajes se basa en la cantidad de mensajes intercambiados entre dos contactos de mensajería y el largo de dichos mensajes. La idea es tratar de identificar cuando se trata de una conversación entre dos personas utilizando estos valores estadísticos. Es decir, las personas escriben a una velocidad que permite estimar el número de mensajes que se pueden escribir y enviar en un tiempo determinado. Si el número de mensajes intercambiados en un intervalo de tiempo es elevado, entonces se sospecha de esa conversación ya que difícilmente dos personas pueden estar escribiendo a esa velocidad. También se puede sospechar de los mensajes muy largos, comparando con el largo promedio de mensajes enviados en conversaciones de mensajería entre las personas.

Implementación.

Para lograr recolectar las estadísticas de todas las conversaciones establecidas se decidió modificar el proceso principal (main) de la aplicación IMSpector. Se creo un modulo para manejar las estadísticas de manera que contenga toda su implementación. El módulo creado esta formado por los archivos stats.h y stats.cpp. En el apéndice 2 se encuentran ambos archivos para mas detalles.

Las estadísticas de mensajes se almacenan en una estructura destinada a esos efectos.

```
typedef struct {
      char from[MAX_LENGTH];
      char to[MAX_LENGTH];
      int nMSGS;
      float avg;
      float len_avg;
      time_t t_start;
} _stats;

typedef struct {
      int tope;
      _stats stats[MAX_STATS];
} stats;    // Arreglo que almacena hasta MAX_STATS conversaciones
            // con sus respectivos datos estadísticos.
```

Y las operaciones utilizadas para mantener dicha estructura actualizada son:

```
/* Crea la estructura para almacenar las conversaciones. */
int create_stats(stats* st);

/* Devuelve el puntero a la conversación entre from y to*/
_stats* find(stats* st, char* from, char* to);

/* Escribe a un archivo de nombre "file_name" las estadísticas
recolectadas en la estructura st */
int write_stats_to_file(stats* st, char* file_name);

/* Crea las estadísticas para la conversación entre from y to y setea
la hora en que fue iniciada la conversacion */
int conv_stats_create(stats* st, char* from, char* to);

/* Setea la cantidad de mensajes intercambiados entre from y to */
int set_count_msg(stats* st, char* from, char* to, int count);

/* Setea el promedio de mensajes por minuto intercambiados entre from
y to */
int set_avg_msg(stats* st, char* from, char* to, float avg);

/* Setea el largo promedio de los mensajes intercambiados entre from y
to */
int set_len_avg_msg(stats* st, char* from, char* to, float len_avg);

/* Obtiene la cantidad de mensajes intercambiados entre from y to */
int get_count_msg(stats* st, char* from, char* to);

/* Obtiene el promedio de mensajes intercambiados entre from y to */
float get_avg_msg(stats* st, char* from, char* to);

/* Obtiene el largo promedio de mensajes intercambiados entre from y
to */
float get_len_avg_msg(stats* st, char* from, char* to);
```

```
/* Obtiene la hora en que fue iniciada la conversación entre from y to
*/
time_t get_time_init_conv(stats* st, char* from, char* to);
```

Como la estructura necesita ser modificada para cada conversación que se establece, y como la aplicación IMSpector utiliza subprocesos para manejar cada conversación, fue necesario colocar esta estructura en un área de memoria compartida, de manera que pueda ser accedida por todos los subprocesos. Es por ello que se utiliza un área de memoria compartida. Se utilizó System V - IPC (inter proccess comunication). Además del área de memoria compartida se utilizaron los semáforos para señalizar el acceso al área de memoria.

5.3. Pruebas realizadas.

A continución se presentan las pruebas realizadas con los diferentes algoritmos de codificación y sus métodos de detección y/o filtrado.

5.3.1. Escenario 1 – IPoIM + IMSpector.

En esta prueba se tienen dos instancias de ipoim comunicándose a través de un firewall con la aplicación IMSpector, tal como muestra la figura 11. En donde las flechas blancas representan la comunicación real, mientras que las flechas negras representan la comunicación establecida en el túnel.

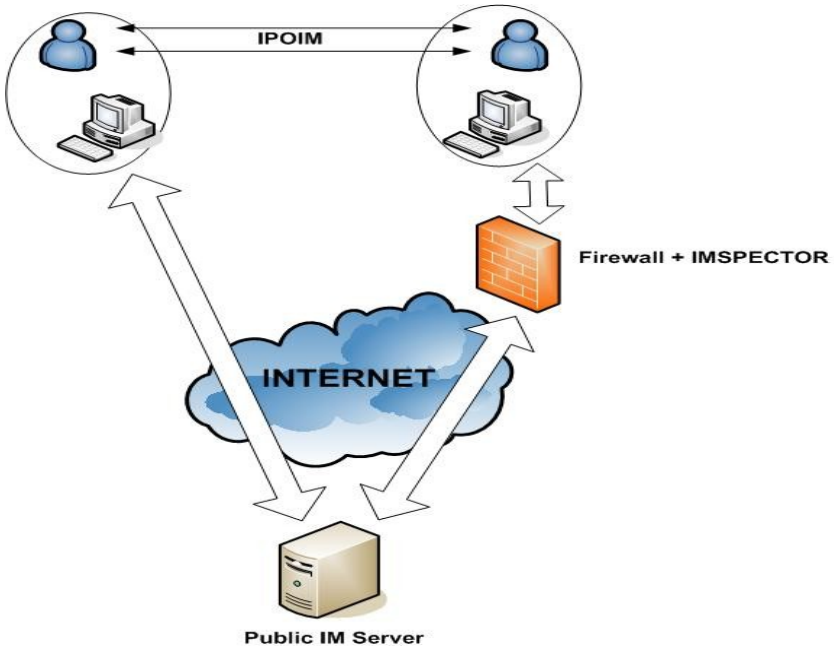

Figura 11: IPoIM + IMSpector

Ejemplo de ejecución #1:

IPoIM: Codificación con Base64. IMSpector: filtro de contenido de base64.

Para este ejemplo se utiliza el método de codificación Base64 para la aplicación IPoIM. Mientras que en la aplicación IMSpector se aplica el filtro de contenido de Base64.

La salida de IMSpector es la siguiente:

```
imspector: Debug: Event: Client address: 192.168.86.140:1544
imspector: Debug: Event: Timestamp: 1192417804
imspector: Debug: Event: Protocol: MSN
imspector: Debug: Event: Type: MSG_OUTGOING
imspector: Debug: Event: LocalID: mlr-ipoim@fing.edu.uy
imspector: Debug: Event: RemoteID: asy-ipoim@fing.edu.uy
imspector: Debug: Event: Data: )
AgAEq38AAAEAAAAAAAAAADwAAABwZmVDX+JtLMdXkNW5qePs0qt5IceVdBx/pHb1T2Zdfs
O6YeZ/4kNPNUswYY0AUZ4HTKEUIAwvVLdfdmg=
imspector: MSN: Got 7 bytes of header
imspector: MSN: Command: ACK
imspector: MSN: Got 91 bytes of header
imspector: MSN: Command: MSG
imspector: MSN: header: MIME-Version value: 1.0
imspector: MSN: header: Content-Type value: text/plain; charset=UTF-8
imspector: MSN: header: User-Agent value: IPOIM/2.0
imspector: MSN: header: X-MMS-IM-Format value: FN=MS%20Sans%20Serif;
EF=; CO=0; PF=0; RL=0
imspector: Base64: filtering before: original: )
AgAEq38AAAEAAAAAAAAADwAAABwZmVDX+JtLMdXkNW5qePs0qt5IceVdBx/pHb1T2Zdfs
O6YeZ/4kNPNUswYY0AUZ4HTKEUIAwvVLdfdmg=
imspector: Base64: filtering after: modified: Se detecto codificacion
Base64**********************************************************
*****
```

Como muestra la salida del IMSpector, cuando se envían datos codificados utilizando Base64, los mismos se reemplazan por asteriscos, y se muestra el mensaje: "Se detectó codificación Base64" al comienzo.

Esta acción además de detectar el pasaje de datos que posiblemente sean codificados utilizando Base64, impide la correcta ejecución de IPoIM ya que modifica los mensajes enviados haciéndolos incomprensibles por IPoIM. Al no ser reconocido como un mensaje correspondiente al túnel IP, el mismo se descarta. Como todos los mensajes de este tipo son modificados, evidentemente son todos descartados por IPoIM, con lo cual la aplicación no puede ejecutarse con normalidad. Por consiguiente el filtro es efectivo para detectar y detener la comunicación entre equipos que utilicen IPoIM como alternativa para establecer un túnel IP.

Ejemplo de ejecución #2:

IPoIM: Codificación con Diccionario. IMSpector: filtro de contenido de base64

Para este ejemplo se utiliza el método de codificación Diccionario para la aplicación IPoIM. Mientras que en la aplicación IMSpector se aplica el filtro de contenido de Base64.

La salida de IMSpector es la siguiente:

```
imspector: Debug: Event: Client address: 192.168.86.130:60159
imspector: Debug: Event: Timestamp: 1192416591
imspector: Debug: Event: Protocol: MSN
imspector: Debug: Event: Type: MSG_INCOMING
imspector: Debug: Event: LocalID: asy-ipoim@fing.edu.uy
imspector: Debug: Event: RemoteID: mlr-ipoim@fing.edu.uy
imspector: Debug: Event: Data: )amoebae accumulators bandwidth polling
include accumulator abbrev acolyte a accumulators abbrevs abbrev
abbrev abbrevs accumulators abbrevs flush abbrev a accumulators
sneakers bytes tweeters cat newbies broken diked diked waldoes
spiffiest mungs snark thuds ding neophilias wormhole sacred snark
cloning warez theory spoofs confuser echo pistol overrun hackers
gnarliest infinities musics robuster dd woofers meta manged zigamorphs
dragons orphans hoardings nukes dike flapped biffed fossil ripping gun
flooding bauds cretinous mundanes pistols brittler cranks snails cons
shell slims nuking cycling buzzing bazaars byte byte wonkiest inflate
frogginging plumbing tweaking biffed mung visionary grues gonking
wormholes root demigods puffing toadding bogus shebangs fan roaching
baroque scrogs admin catatonic trap flowchart ding execked golden
cowboy
imspector: MSN: Got 92 bytes of header
imspector: MSN: Command: MSG
imspector: MSN: header: MIME-Version value: 1.0
imspector: MSN: header: Content-Type value: text/plain; charset=UTF-8
imspector: MSN: header: User-Agent value: IPOIM/2.0
imspector: MSN: header: X-MMS-IM-Format value: FN=MS%20Sans%20Serif;
EF=; CO=0; PF=0; RL=0
imspector: Base64: filtering before: original: )alts abbrevs avatar
pop incing abbrev abbrev acolyte accumulator abbrev accumulators
abbrevs abbrevs abbrevs accumulators abbrev flytrap abbrev abbrev
abbrevs footprints mutters scags trawls rip emailed flap polling
spikes gonks donutses notworks fascist losers visionaries supports
lobotomy cubing dodgier managements supports whacking crashes cascades
codes cube scribble poll bumped zip orthogonal flamed spams burbles
calculator cyberpunks hacked playpens poke blats faultiest byte worms
flood bum quads cans cracking flaky blocking flaps filter sacred email
thumbs baud block cranks clobbering h grokked golden followups voiced
condoms flakiest netter payware brittle segmenting quad muttered
crocks accumulator shebangs engines virus lobotomy coasters
```

hyperspaces tee life hose decays nanobot zapped zip toggled glasses benchmarks downing flapped

imspector: Base64: filtering after: modified:)alts abbrevs avatar pop incing abbrev abbrev acolyte accumulator abbrev accumulators abbrevs abbrevs abbrevs accumulators abbrev flytrap abbrev abbrev abbrevs footprints mutters scags trawls rip emailed flap polling spikes gonks donutses notworks fascist losers visionaries supports lobotomy cubing dodgier managements supports whacking crashes cascades codes cube scribble poll bumped zip orthogonal flamed spams burbles calculator cyberpunks hacked playpens poke blats faultiest byte worms flood bum quads cans cracking flaky blocking flaps filter sacred email thumbs baud block cranks clobbering h grokked golden followups voiced condoms flakiest netter payware brittle segmenting quad muttered crocks accumulator shebangs engines virus lobotomy coasters hyperspaces tee life hose decays nanobot zapped zip toggled glasses benchmarks downing flapped

Como se puede apreciar, el filtro no detectó codificación Base64 (lo cual es esperable, ya que no se trata de una codificación Base64) y por lo tanto no se realizó acción alguna. De esta manera, y con este tipo de codificación se logra evadir filtros de contenido que busquen algún patrón, como la codificación en Base64. Para evitar la comunicación utilizando este tipo de codificación es necesario utilizar otra técnica.

Ejemplo de ejecución #3:

IPoIM: Codificación con Base64. IMSpector: Recolección de estadísticas de uso.

Para este ejemplo se utiliza el método de codificación Base64 para la aplicación IPoIM. Mientras que en la aplicación IMSpector se recogen estadísticas de uso. Se obtienen datos como el número de mensajes intercambiados, el promedio de mensajes por minuto y el largo promedio de los mensajes.

Las estadísticas recogidas por IMSpector son:

from:	to:	numero de mensajes:	prom mensajes: (mens/minuto)	largo promedio de mensaje:
xxxx@fing.edu.uy	xxxx@fing.edu.uy	299	149.50	193.01
xxxx@fing.edu.uy	xxxx@fing.edu.uy	148	148.00	193.05
xxxx@fing.edu.uy	xxxx@fing.edu.uy	1312	131.00	193.01
xxxx@fing.edu.uy	xxxx@fing.edu.uy	10569	120.10	193.01
xxxx@fing.edu.uy	xxxx@fing.edu.uy	3154	108.76	192.79

De las estadísticas recogidas se obtiene que el largo promedio de los mensajes enviados fue de unos 193 caracteres por mensaje, y que la cantidad de mensajes enviados por minuto asciende a unos 140 mensajes por minuto promedialmente.

Mientras que las estadísticas recogidas por IMSpector para conversaciones reales entre personas fueron las siguientes:

from:	to:	numero de mensajes:	prom mensajes: (mens/minuto)	largo promedio de mensaje:
xxxx@hotmail.com	xxxx@hotmail.com	135	3.75	18.55
xxxx@hotmail.com	xxxx@hotmail.com	271	7.74	22.49
xxxx@hotmail.com	xxxx@hotmail.com	120	3.53	27.72
xxxx@hotmail.com	xxxx@hotmail.com	40	3.64	18.57

Donde se aprecia que el largo promedio de los mensajes es de unos 22 caracteres por mensaje, y que está muy lejos de 193 caracteres. Además el promedio de mensajes por minuto también es un número razonable promediando los 5 mensajes por minuto.

Ejemplo de ejecución #4:

IPoIM: Codificación con Diccionario. IMSpector: Recolección de estadísticas de uso.

Para este ejemplo se utiliza el método de codificación Diccionario para la apliación IPoIM. Mientras que en la apliación IMSpector se recogen estadísticas de uso como en el ejemplo de ejecución #3.

Las estadísticas recogidas por IMSpector son:

from:	to:	numero de mensajes:	prom mensajes: (mens/minuto)	largo promedio de mensaje:
xxxx@fing.edu.uy	xxxx@fing.edu.uy	3434	245.29	546.09
xxxx@fing.edu.uy	xxxx@fing.edu.uy	1445	289.00	575.99
xxxx@fing.edu.uy	xxxx@fing.edu.uy	3336	256.62	560.64

Nuevamente, como en el ejemplo anterior, los datos estadísticos recogidos están muy lejos de los valores normales para conversaciones de mensajería instantánea entre personas. Esta vez el largo promedio de mensajes enviados es del entorno de 550 caracteres por mensajes. Esto es completamente inusual en conversaciones reales. Por otro lado el promedio de mensajes enviados por minuto fue del entorno de 250 mensajes. Este promedio implica que se enviaron 4 mensajes por segundo, y con un largo promedio de 550 caracteres. Es decir unos 2200 caracteres enviados por segundo.

Teniendo en cuenta que una persona es capaz de escribir unas 100-120 palabras por minuto en los mejores casos, es decir unas 2 palabras por segundo, y que el largo promedio de las palabras es del entorno de 8 caracteres, entonces una persona sería capaz de generar unos 16 caracteres por segundo.

De los resultados anteriores se puede determinar que con esos valores estadísticos es imposible que se trate de conversaciones reales entre personas. Por este motivo, la técnica de detección utilizando estadísticas de mensajes es efectiva para determinar la existencia de un uso indebido de la mensajería instantánea. Con ésta se pueden tomar medidas para impedir el uso incorrecto de las herramientas de mensajería.

5.3.2. Escenario 2 – IPoIM + IM SecurityNET.

En este escenario se tienen las dos instancias de IPoIM comunicándose a través de un servidor de filtrado de contenido de la empresa Toolsthatwork. El producto de filtrado es IM SecurityNET™. Aquí otra vez las fechas representan la comunicación tal como se explicó en el escenario anterior.

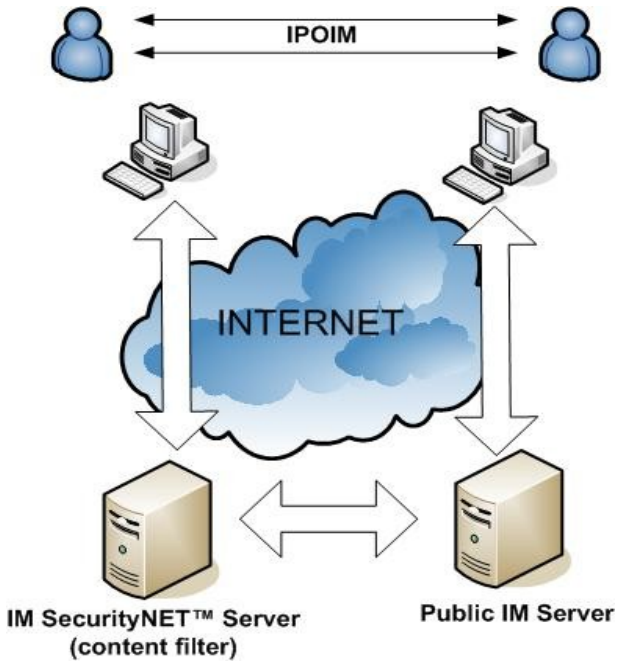

Figura 12: IPoIM + IM Security Net

La comunicación se establece tal como muestra la figura 12. Es decir, cuando una de las instancias de IPoIM envía un mensaje a la otra, éste es enviado al servidor de IM SecurityNET, luego es sometido al filtro de contenido y es enviado al servidor público de mensajería. Por último el servidor de mensajería envía el mensaje al destinatario.

Este producto tiene capacidades de filtrado de contenido en base a una lista de palabras o frases que se desean controlar. Por esta razón fue elegido para realizar pruebas ya que nos da la posibilidad de realizar varias pruebas a fin de determinar si es capaz de bloquear el tráfico generado por IPoIM.

La lista de palabras tiene dos modalidades. Se puede utilizar como lista de palabras permitidas, o como lista de palabras no permitidas (black list). En el primer caso todas se permiten, y en consecuencia, si se envían palabras que no están en la lista, éstas son bloqueadas. En el segundo caso todas se bloquean, con lo cual, cada palabra enviada se verifica contra la lista, y en caso de encontrar una coincidencia, la palabra es bloqueada, en otro caso se permite. No se pueden combinar algunas palabras permitidas con algunas no permitidas.

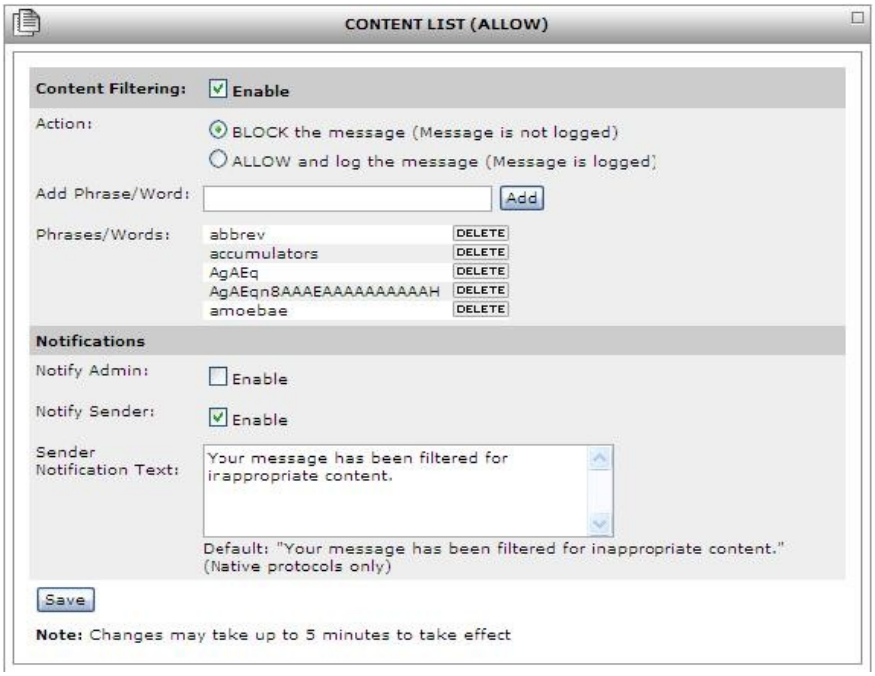

Figura 13: Interfaz de configuración de IM Security Net

En la figura 13 se muestra la interfaz de configuración de la aplicación:

En la misma se puede apreciar que está configurado el filtro de contenido, y que se especifica una lista de palabras no permitidas:

abbrev
accumulators
AqAEq
AqAEqn8AAAEAAAAAAAAAH

Ejemplo de ejecución #5:

IPoIM: Codificación con Base64. IM SecurityNET: Filtro de palabras mediante "black lists".

En este ejemplo se utiliza el método de codificación Base64 y el tráfico es ruteado a través del servidor de filtrado de contenido de IM SecurityNET. A su vez se configuró un filtro de contenido con las siguientes palabras no permitidas:

AqAEq

AqAEqn8AAAEAAAAAAAAAAH

Lo que se busco con estas "palabras" fue bloquear el tráfico de datos codificados en Base64. Según pruebas realizadas, se notó que todos los mensajes llevan esas "palabras" o secuencia de caracteres al comienzo. La idea es ver si con esto es suficiente para bloquear el tráfico IPoIM.

La ejecución de IPoIM con este filtro de contenido activo dio como resultado:

```
mensaje                                                                    ->
AgAEqn8AAAEAAAAAAAAAAHwAAAA/Ff3rFTgqSsPIbeuyvyfAQAnPLFx2y/sNRcqIbSZ+vt
z/UQHoeY6YivqyHqp5EBl9ltUAf2ZXtOY1QRAOPbSNdjgow82aZ3bgXoLctwiXNz44N+Wp
6w4gl+zm2L6cFmo3r0jJmN79MgbJHt8NrW07W/PC5j/PHPyFUUE5
IM ajeno a VPN! Descartado... IM Administrator: Your message has
been filtered for inappropriate content.. code=I
mensaje                                                                    ->
AgAEqn8AAAEAAAAAAAAAAHwAAAC03dhPMeywc1mQpRyAkQUbL9vICLCi3k3Nw1A8aMFUe0
Z1bB1p5fcCA0ChKWFMgmMUFhp4xmz9tUz6Y04s2vokS68lWHc1N0FFPqMlRi+Ilfg/sYLb
izN2r66R7IdpZBIynDj9LLAWfTaIvmD0QfGmw8P8BIpoAl+9vCu8
mensaje                                                                    ->
AgAEqn8AAAEAAAAAAAAAAHwAAAD3CzMdW6fY9CwVpjhwqrUC3Hav9wcKhxemY2n39GOmVc
G5pUR+Yscqc6XFCi46LrQYaQBKwTlnEcvtDRQXAEqmmlxngZFKLgjhT3iMyyVa0s4Q6yeF
e/Af+CFuj1HPh2ecTvQGo1sOktC/+qOCnluxZUXG3ZnefwytcOi8
IM ajeno a VPN! Descartado... IM Administrator: Your message has
been filtered for inappropriate content.. code=I
mensaje                                                                    ->
AgAEqn8AAAEAAAAAAAAAAHwAAABTidbr4TXbdyYa7zidTt07obfHb4dqYSwqlyL92vYPse
ZaRDwnfCSCig202BxwbZhB48bfF5GiJQi1/s+xCh4Jo5a37pzTklL4D8OVss/qDpFVElO8
R4W86j6t2X815jdUG87JgTh9bAXRVUjoxkIYyBb0npWsBx4Hhepk
mensaje                                                                    ->
AgAEqn8AAAEAAAAAAAAAAHwAAAAuCiPtdrkwxaZAX2Fao3nrzp4ysfyGygNRTLZALr+tv1
czUlBuCImbKwZuWFM+4kP2XUAs6wRnyznU45MQla3P52w5FZIwJju+bq05TOpZtUvaV9bq
XhCkg1Q+1Scd4l1Rs8y9oqM6LttnX5WXnMRZPYBsWSciXMQuofxF
mensaje                                                                    ->
AgAEqn8AAAEAAAAAAAAAAHwAAAA1nhcGAKsFuaAEh+z2k26mZ4Uex7BRfIr1E7WFlBKwH/
VtmboEYD6Q5XzW5B8ycEeQ+CkwGfbgml7NCx/5rGuYBOfzUYRUkeAEccgQvrB4Y0O+dRfM
rhbdUBTc8wpgRtOv1S3hEtKC47k9YYEghXeI3z92UUzShsuixRQ5
IM ajeno a VPN! Descartado... IM Administrator: Your message has
been filtered for inappropriate content.. code=I
mensaje                                                                    ->
AgAEqn8AAAEAAAAAAAAAAHwAAAATGHDojktZP9ugPiFPLdJD9MYKtvTRFFmsVYu/qlmOkO
8grezooUAVEmcl7U5xlcDln0dQA27xFkBCMIu/F7Ek4fRGX+KXNPBLYIpQMvd6mYyddRhv
Kp5s+osMwRH3+DxipJxXVYvqf0qKIL/SKXSC0NEjAShrBdsy7y01
mensaje                                                                    ->
AgAEqn8AAAEAAAAAAAAAAHwAAAD2ha3Ck5rMUfLtpN4O6n4Tp1XlSSr0z19X0uFDNFjkxk
dmcps8VsYsTzWFPkDBBiaDVhgYhc6F6Q1JWspRdfIana+s2HYEWB+aKXMdCpcfEnmFaYX4
e8nusx0HEA31n3XpzjcFz2Y6LThmAINdyFnQY4Jqazk+MBvcG9i3
```

IM ajeno a VPN! Descartado... **IM Administrator: Your message has been filtered for inappropriate content**.. code=I

De manera que como se ve en la salida del programa, el filtro de contenido fue efectivo bloqueando los mensajes enviados entre las instancias de IPoIM. Se aprecia el mensaje de notificación (que se puede configurar) que informa a los usuarios de mensajería que el mensaje ha sido filtrado. "IM Administrator: Your message has been filtered for inappropriate content"

Evidentemente en estas condiciones el túnel IP no se pudo establecer ya que los mensajes fueron bloqueados por la aplicación. Cabe señalar que esta acción podría ser tomada por un administrador de mensajería instantánea una vez revisados los logs, y detectado el tráfico codificado en Base64. Es decir que se debería realizar un análisis de los logs de mensajería en busca de conversaciones que no parezcan ser mantenidas entre seres humanos.

Si bien este filtro fue efectivo, la finalidad de estos filtros de contenido es evitar ciertas palabras ofensivas, o quizás la fuga de información, pero siempre se configuran con palabras válidas de algún idioma hablado. Por esta razón la probabilidad de evadir un filtro de estas características con IPoIM es elevada, a menos que el administrador haya configurado alguna palabra que intente buscar algún patrón (como lo hicimos en este caso, buscando un patrón de codificación en Base64)

Ejemplo de ejecución #6:

IPoIM: Codificación con Diccionario. IM SecurityNET: Filtro de palabras mediante "black lists".

En este ejemplo se utiliza el método de codificación Diccionario y el tráfico es ruteado a través del servidor de filtrado de contenido de IM SecurityNET™. En este caso se configuró un filtro de contenido con las siguientes palabras no permitidas:

abbrev
accumulators

Con esta lista de palabras se intenta bloquear con éxito el tráfico de IPoIM. Estas palabras pertenecen al diccionario utilizado en la codificación Diccionario.

La ejecución de IPoIM con este filtro de contenido activo dio como resultado:

```
mensaje -> amoebae abbrev bandwidth poll inced accumulator accumulator
admins a abbrevs abbrevs abbrevs abbrevs abbrev abbrevs abbrev diddled
abbrevs abbrevs a accumulators server orphans snarf lobotomy grok
nanobot boinking wart kits said bits footprints diffed crayons groks
blinks alts munging generated biff diddle kit snapping grok leech
rudest smurf meg compos blink grovels cleaner munched cyberpunk smokes
quads screens cascades nickle admin twink fingering urchin trapped
```

cycled eds chromes demigod crept meg toasted kludged flamer snarfing quine inflates abbrev grokked freeze

IM ajeno a VPN! Descartado... **IM Administrator: Your message has been filtered for inappropriate content**.. code=I

mensaje -> amoebae abbrev bandwidths polling inced accumulator accumulators admin abbrev a accumulator accumulator abbrevs accumulator abbrev abbrev deprecated a abbrev abbrevs musics deprecated urchin check zigamorph inflating fudge frog deltas tensest swabs trawls chompers zombies bumped chugs deadlock warts zapped rogues acolytes diddles saying accumulator priesthoods wizard dike evilest byte broke studliest earthquakes cubinging popping modding warez bounced catatonic dongles tick kit glasses frog clobbering monstrosities hexadecimals samizdat generated glasses thrash spell snarf chads bible footprint trivial deprecated toggled spammed studliest

IM ajeno a VPN! Descartado... **IM Administrator: Your message has been filtered for inappropriate content**.. code=I

mensaje -> amoebae abbrev atomic pop included a abbrevs acolytes a abbrev accumulator abbrevs accumulators a abbrev accumulators deprecated accumulators a accumulators cretinous visionary tooled generated bummed clobbered clobber barf farmings cleaner conses thunk bummed bytes troglodyte hop numberses flaps wormholes nature thunk nonlinear zapping hacks iron voicing trawled strudel cookbooks flatten fora leech mockingbirds blivet grovelling hops trapping walk waldoes hacking uploading snarfs boot fool cretin meta hairballs dodgy recursions twiddle hang bomb macros catting bandwidth spell selvages slab gens barf

IM ajeno a VPN! Descartado... **IM Administrator: Your message has been filtered for inappropriate content**.. code=I

mensaje -> amoeba abbrevs asbestos polls included accumulator abbrevs alt accumulator abbrevs abbrevs abbrev abbrev a accumulator accumulators deprecated a accumulator abbrev monstrosities snapped flamers lint mangeds kludged mailbombs scrogs diffed enhancement wired gun zombies raped zigamorph voice incs deltas biffs golden memes tronned ass frying softy virus wall board linted pings snarfing happily fences slab zips warez management canned tronned chars catting meg interesting flakiest slack admins cubes foregrounded deckle donutses boxes nanobot ices crunch saying mangeds linted biff chug minifloppies

IM ajeno a VPN! Descartado... **IM Administrator: Your message has been filtered for inappropriate content**.. code=I

Como se puede ver, nuevamente el filtro fue exitoso ya que impidió la comunicación con IPoIM. Sin embargo, no es real tener un filtro de contenido con palabras validas que no sean ofensivas ni estén violando ninguna norma de uso. En este caso el filtro estaba formado por palabras válidas del idioma inglés. Palabras que perfectamente pueden ser utilizadas en conversaciones reales, sin que su uso implique un uso incorrecto de la mensajería instantánea. Un filtro de estas características impide el uso normal de las aplicaciones de mensajería. Esto determina que un filtro así no es adecuado para brindar seguridad al uso de las herramientas de mensajería instantánea.

5.3.3. Escenario 3 – IPoIM + Symantec IM Manager.

En este escenario representado en la figura 14 se tienen las dos instancias de IPoIM comunicándose a través de un servidor de filtrado de contenido que tiene instalada la aplicación Symantec IM Manager. Nuevamente las fechas representan la comunicación tal como se explicó en los escenarios anteriores.

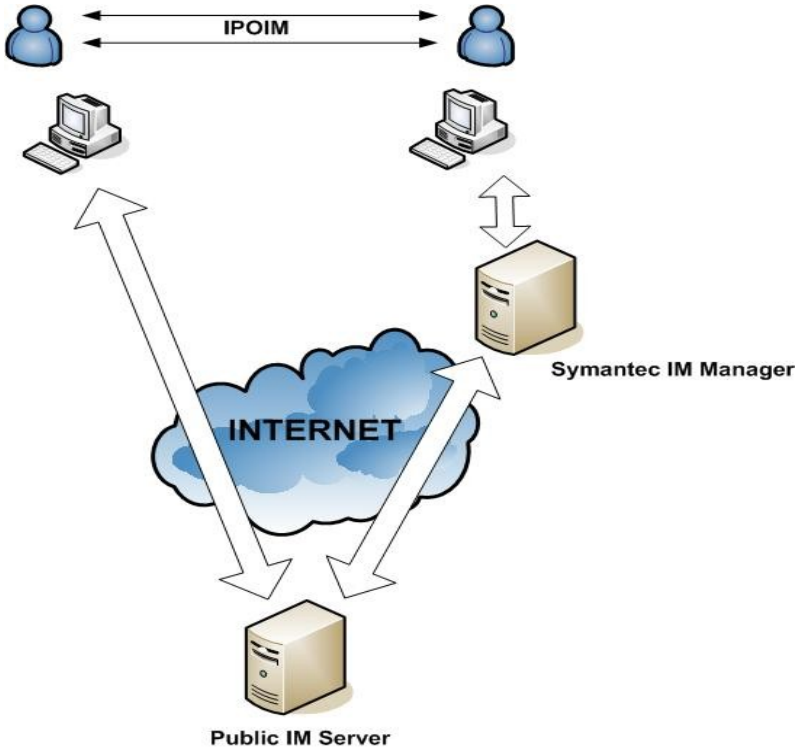

Figura 14: IPoIM + Symantec IM Manager

La aplicación Symatenc IM Manager permite configurar entre otras cosas una lista de palabras que se desean filtrar. Estas palabras pueden ser palabras exactas, o bien fragmentos de palabras. Se pueden filtrar las palabras que empiecen con determinados caracteres, o que finalicen con otros, etc. También se pueden especificar expresiones regulares.

Se puede ver un ejemplo en la figura 15:

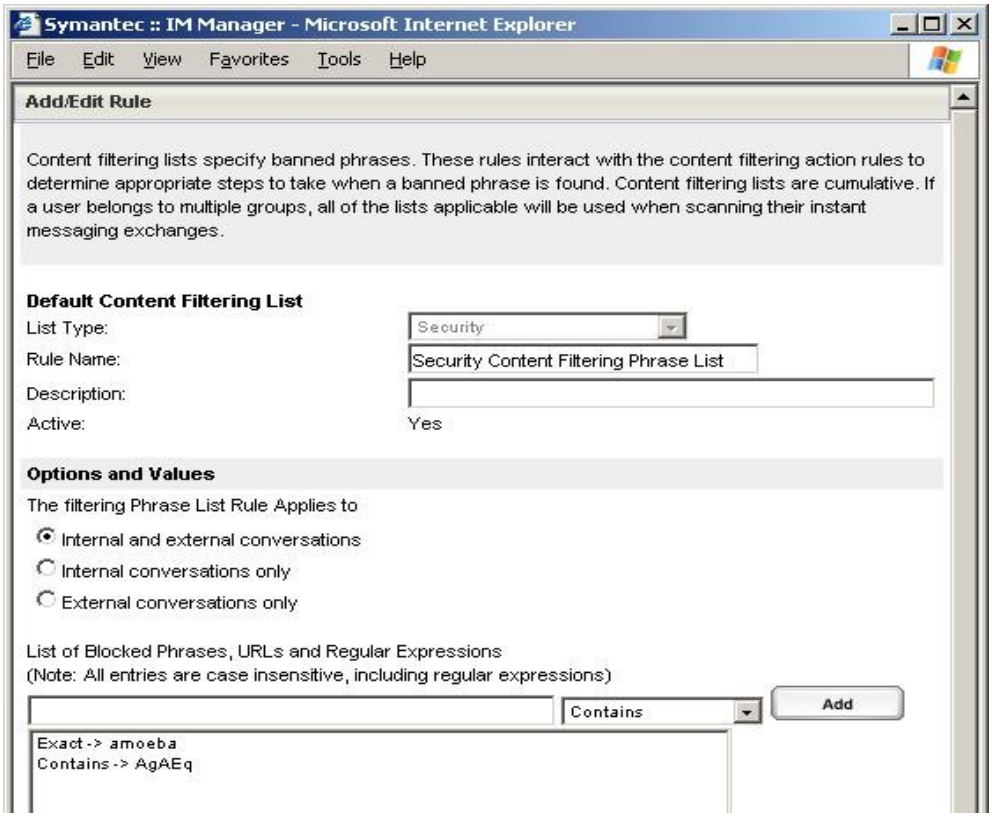

Figura 15: Interfaz de configuración de Symantec IM Manager

Ejemplo de ejecución #7:

IPoIM: Codificación con Base64. Symantec IM Manager: Filtro de palabras mediante "black lists".

En este ejemplo se utiliza el método de codificación Base64 y el tráfico es ruteado a través del servidor que tiene instalada la aplicación Symantec IM Manager.

Para este ejemplo se agregó la siguiente expresión para que la aplicación la filtre:

Contains-> AgAEq

Es decir, se busca filtrar todos los mensajes que contienen en alguna parte la secuencia de caracteres "AgAEq". Esto se basa en que todos los mensajes que son generados por IPoIM y son codificados en Base64 comienzan con los mismos caracteres. (Se determinó empíricamente).

La ejecución de IPoIM con este filtro de contenido activo dio como resultado:

mensaje ->
AgAEqn8AAAEAAAAAAAAAAHwAAAAO5uw8U/E4Z2Pm4spwLU0RJxYvYkYWYCYPGQbJunIzbo
rUolXEaUrgWjugmPUuBa2atJkhHxE5XRnXMFUJfnvtz7ng8hwU16Gq5D0LKKnO5rAAbbPl
i/LqT9L+0adhM1Cvq5dKY3EaZ6iXM1dLHISmOeeWqYeja+mZG1Rw

mensaje ->
AgAEqn8AAAEAAAAAAAAAAHwAAADWZIt1eaV0atze6IivHajduTM8NZGzLjcmFMGVPyn9YW
jrxA0WMiWLLcfHiWtdafXg90jTceFZXuDa5/ChhgrhFzN1B0Ega/cqu6q7ua2N593xpLeV
nhItPBgyhIpBsYksK9EWKKvvDZ7gA3ALJ/Zv5CywOvUZlTLib0Ny

IM ajeno a VPN! Descartado... ***IM Administrator: Your message has been intercepted by Symantec IM Manager.*** code=I

mensaje ->
AgAEqn8AAAEAAAAAAAAAAHwAAADKT2GVTb45pDdD4qOGICycZcGjv1/AwVGyQI45TSWrYT
/hGyjeCUHBBe99pXguAAlmUAlymesrw1e03/dY/N6Ktip92mtvUYwmmtlToHcWKOXBvDZM
6+y9RhDCHyj08ai+C05fSFVYrVAo9uxBaK8bwKckUIcq9zovRyqk

mensaje ->
AgAEqn8AAAEAAAAAAAAAAHwAAAAkNGYqd6enLN6DXcrtvrSPt2k+1z6dDe2bKH2TxJpeK0
b6mKwWgpkGj2xWZDNCUXwLzXIOj10oqHY5k32lfw8NJ0DZebrRzzGJ4Z6yYHkgiEM48jjf
qIM7UrIdJT470k4LUk3G6gqs5ENsmQQ8yoqvvZthVgpNK5GDwm6j

mensaje ->
AgAEqn8AAAEAAAAAAAAAAHwAAADQhRhorNDQqx50ewGEKZoZgtr6Y+VbMA2JCLljlAAauz
sFIZd/r7LNrldUMBGYY4nkUZOTTpUQinpVqB9fLhKshkgl9etUt0ycKYrplycLM6U9R+F2
VBGgmxMebx1++MX/r4tu6Oz7OwcS1Us63NSjiANj2H2K33+RPCgX

IM ajeno a VPN! Descartado... ***IM Administrator: Your message has been intercepted by Symantec IM Manager.*** code=I

mensaje ->
AgAEqn8AAAEAAAAAAAAAAHwAAAC51VjrEgnXrL3LEPtyRS8y1BypMuuudhooP7QwBT2/Ai
2ov7vYlBS7YYZaUnDqqYD+5OjvW4M2n4vUzpPEcJyam92vZCj9oW3tEjsT46jvVH24QGTY
ctf75SyF8vzSvuZ9R5vvTxKZoMGhfkJFmCs6EtUs2/AsZRmT4FZM

mensaje ->
AgAEqn8AAAEAAAAAAAAAAHwAAAAuH54E8iNBThRowEPJQJGh68Gj9imTn7Ud5JQCOpW/6W
IEFA1eZMiCW9fFSJWnbF2W0KUNzEKCo8Ku2VPoaKCzayc+8N0kwhIpKBxggB98s3yK0ezp
/GuVGMCyM8pCYKCwXvKGuaYtWiJ9CvfzXMEgP/UiF9bPRcQYN1A8

IM ajeno a VPN! Descartado... ***IM Administrator: Your message has been intercepted by Symantec IM Manager.*** code=I

mensaje ->
AgAEqn8AAAEAAAAAAAAAAHwAAAAvUtZJpMbM5d0LGx+9KZ8MklpHzSZ5ZjYK1FlV39eE0E
xKLWmlzMWBIoio6i9/eR7pmyBOL3BjUHbYKaopjDnQvn91QGJj4+K0kML+NKgeT5e4I9Sn
4N++hngivhQkR6bg442N8caLswBl2IKyGMPhO3pnSbtTpeoJCetX

mensaje ->
AgAEqn8AAAEAAAAAAAAAAHwAAABWc6dJ3M1A57BNep39NL5EYNekyCjDHo5br/2fi5EkNv
912BrjcdGq0W5I4zaHpVzknw6gNpOj/d8BSzuobe1OPUE1uqVi3vMOstv/cmTRP0hEPelz
UQby1PxXUk3fwnL6ruLg0n3XIth4doKz2+NNRlxXtFxbBJ6Mt3vj

mensaje ->
AgAEqn8AAAEAAAAAAAAAAHwAAAALx/vIUWB5EiUk5fP3CmoNg9Po/ihNqOmx2RqOLUYU1e
ffc04dsL6e5yQMKEQDVOFv6D9Gcomcx8VIavKDizx515vt30/vPoD1SW6VWWW9TbvbF6Tu
ab3ykanhkHOZD0Ma1xjRDIHJ4tjY8w+T3gyctXBlovnpDCDC2PiQ

mensaje ->
AgAEqn8AAAEAAAAAAAAAAHwAAACYR0V7pMGneuOx/HukgbBaav4IMRdWykDCEiFfdl1wpd
wM+qR9QH9RZVbX5Jl/OQn38UQs6/XIyrepxW7FqNd77wrA+sMseTkDmpsUhUHumPNrWXJG
cKgGWxZAM0hnIL31Pb7sakwrWEyJp0uKcgtvBkDTDOpnHXx7hWMf

IM ajeno a VPN! Descartado... ***IM Administrator: Your message has been intercepted by Symantec IM Manager.*** code=I

Como se ve en la salida del programa, los mensajes son filtrados por Symantec IM Manager, siendo el usuario notificado con el siguiente mensaje "***IM Administrator: Your message has been intercepted by Symantec IM Manager.***"

De esta manera, logramos filtrar el tráfico generado por IPoIM. De nuevo en este caso se utilizó un patrón que sabemos que se presenta en las codificaciones en Base64. De utilizar solo palabras, no se podría haber filtrado dicho tráfico.

Ejemplo de ejecución #8:

IPoIM: Codificación con Diccionario. Symantec IM Manager: Filtro de palabras mediante "black lists".

En este ejemplo se utiliza el método de codificación Diccionario y el tráfico es ruteado a través del servidor que tiene instalada la aplicación Symantec IM Manager.

Para este ejemplo se agregaron las siguientes expresiones para que la aplicación la filtre:

Exact-> accumulators

Exact-> amoebas

Es decir, se busca filtrar todos los mensajes que contengan alguna de estas palabras: "accumulators" o "amoebas". Dichas palabras son filtradas justamente para impedir el correcto funcionamiento de IPoIM.

La ejecución de IPoIM con este filtro de contenido activo dio como resultado:

```
mensaje -> amoebae abbrev bandwidth poll inced accumulator accumulator
admins a abbrevs abbrevs abbrevs abbrevs abbrev abbrevs abbrev diddled
abbrevs abbrevs a accumulators server orphans snarf lobotomy grok
nanobot boinking wart kits said bits footprints diffed crayons groks
blinks alts munging generated biff diddle kit snapping grok leech
rudest smurf meg compos blink grovels cleaner munched cyberpunk smokes
quads screens cascades nickle admin twink fingering urchin trapped
cycled eds chromes demigod crept meg toasted kludged flamer snarfing
quine inflates abbrev grokked freeze
IM ajeno a VPN! Descartado... IM Administrator: Your message has
been intercepted by Symantec IM Manager. code=I
mensaje -> amoebae abbrev bandwidths polling inced accumulator
accumulators admin abbrev a accumulator accumulator abbrevs
accumulator abbrev abbrev deprecated a abbrev abbrevs musics
deprecated urchin check zigamorph inflating fudge frog deltas tensest
swabs trawls chompers zombies bumped chugs deadlock warts zapped
rogues acolytes diddles saying accumulator priesthoods wizard dike
evilest byte broke studliest earthquakes cubinging popping modding
warez bounced catatonic dongles tick kit glasses frog clobbering
```

monstrosities hexadecimals samizdat generated glasses thrash spell snarf chads bible footprint trivial deprecated toggled spammed studliest

IM ajeno a VPN! Descartado... ***IM Administrator: Your message has been intercepted by Symantec IM Manager.*** code=I

mensaje -> amoebae abbrev atomic pop included a abbrevs acolytes a abbrev accumulator abbrevs accumulators a abbrev accumulators deprecated accumulators a accumulators cretinous visionary tooled generated bummed clobbered clobber barf farmings cleaner conses thunk bummed bytes troglodyte hop numberses flaps wormholes nature thunk nonlinear zapping hacks iron voicing trawled strudel cookbooks flatten fora leech mockingbirds blivet grovelling hops trapping walk waldoes hacking uploading snarfs boot fool cretin meta hairballs dodgy recursions twiddle hang bomb macros catting bandwidth spell selvages slab gens barf

IM ajeno a VPN! Descartado... ***IM Administrator: Your message has been intercepted by Symantec IM Manager.*** code=I

mensaje -> amoeba abbrevs asbestos polls included accumulator abbrevs alt accumulator abbrevs abbrevs abbrev abbrev a accumulator accumulators deprecated a accumulator abbrev monstrosities snapped flamers lint mangeds kludged mailbombs scrogs diffed enhancement wired gun zombies raped zigamorph voice incs deltas biffs golden memes tronned ass frying softy virus wall board linted pings snarfing happily fences slab zips warez management canned tronned chars catting meg interesting flakiest slack admins cubes foregrounded deckle donutses boxes nanobot ices crunch saying mangeds linted biff chug minifloppies

IM ajeno a VPN! Descartado... ***IM Administrator: Your message has been intercepted by Symantec IM Manager.*** code=I

Nuevamente el tráfico fue detenido. Sin embargo, y como sucedió en el ejemplo #6 el tráfico fue detenido porque se especificaron palabras que estaban siendo utilizadas por la codificación con diccionario de IPoIM. El problema aquí es que estas palabras no son contenido ofensivo, y no tienen por que estar filtradas. Son palabras cualesquiera del idioma inglés. Si estuviesen siendo filtradas estas palabras, seguramente no se podría establecer una conversación de mensajería debido a estas restricciones. Además, ¿Con qué criterio se filtrarían dichas palabras? Filtrar palabras válidas no es una buena práctica ya que termina impidiendo el uso normal de las aplicaciones de mensajería. Utilizando la codificación con diccionario se hace muy difícil poder filtrar el tráfico generado por IPoIM.

5.4. Conclusión.

El filtrado de mensajería es una característica presente en la mayoría de las aplicaciones de gestión de mensajería instantánea. La misma permite filtrar palabras o secuencias de caracteres incluídas en los mensajes instantáneos. Con esta característica se pudo filtrar exitosamente el tráfico generado por IPoIM utilizando codificación Base64, así como el tráfico codificado utilizando el diccionario de palabras.

El éxito de los filtros se debió a que las palabras especificadas para el filtrado fueron elegidas de manera de facilitar la detección. En el caso de Base64 se utilizó un patrón que se daba en todos los mensajes. Mientras que en el caso del diccionario de palabras se utilizaron palabras del mismo diccionario.

Sin embargo, en un caso real, una organización que implemente una solución de control y filtrado de mensajería instantánea especificaría como palabras a filtrar las que considere inapropiadas, o que signifiquen la fuga de información de la empresa. Por lo que en principio, no se contemplaría el caso de Base64 por ejemplo, ya que no necesariamente se va a filtrar un patrón, a menos que se detecte dicho tráfico en los logs, y se tome un acción al respecto. Del mismo modo, y teniendo en cuenta a la codificación con diccionario, no se filtrarían palabras validas del idioma inglés ya que estas no significan ninguna amenaza para una empresa.

Por otro lado la recolección de estadísticas de mensajes también es una buena alternativa para detectar un uso indebido de las aplicaciones de mensajería instantánea. Esta característica esta presente en muchas de las aplicaciones de gestión de mensajería instantánea.

Se pudieron determinar los valores razonables para la cantidad de mensajes instantáneos intercambiados, así como el largo promedio de los mismos, cuando se trata de conversaciones entre seres humanos. En base a ello, se puede determinar cuando se esta utilizando la mensajería para otros usos que no son para los cuales fue diseñada la aplicación.

En conclusión, IPoIM puede ser detectado si te toman las precauciones necesarias, pero no es tan fácil si se utilizan técnicas de codificación como el diccionario de palabras, y solamente se filtra mediante listas de palabras. Con codificaciones como Base64 es más fácil filtrar el tráfico ya que las listas de palabras la pueden detener (utilizando patrones). Para ambos tipos de codificación, las estadísticas de mensajes son una herramienta valiosa capaz de detectar (pero no filtrar) el tráfico generado por IPoIM, o cualquier otra aplicación, que no esté haciendo un uso correcto del canal de comunicación de la mensajería instantánea.

6. Conclusiones.

6.1. Resumen del trabajo.

En este proyecto se logró estabilizar el prototipo construído en el proyecto de grado del año 2005. Se realizaron varias modificaciones al mismo brindándole en primera instancia un instalador similar al de otros productos Open/Free. Se migró el prototipo a las nuevas versiones de los productos utilizados en ese entonces, de manera de incorporar los cambios realizados en ambos productos (Pidgin y OpenVPN). Se eliminaron los errores no sistemáticos conocidos que se presentaban en el prototipo anterior, los cuales impedían la correcta ejecución en algunas oportunidades. Hoy en día se puede instalar el prototipo siguiendo los pasos muy simples de instalación, y ejecutar la aplicación sin mayor esfuerzo.

Por otro lado, se reafirmó la posibilidad de establecer túneles IP sobre los protocolos de mensajería instantánea, y más precisamente sobre los servicios públicos de mensajería instantánea. Este hecho no deja de sorprendernos ya que a casi tres años de haber comenzado con estas pruebas, los servicios públicos de mensajería no han filtrado de alguna manera este tipo de tráfico. Este hecho refleja aún más los problemas de seguridad que tienen este tipo de aplicaciones, ya que no se controla de alguna manera la información que se transmite a través de este medio de comunicación.

Se realizó un estudio de impacto en la seguridad de las empresas del uso de la mensajería instantánea como medio de comunicación para el trabajo. Este estudio demostró que si bien la mensajería instantánea es un mecanismo de comunicación y colaboración muy valioso para el desarrollo de las empresas, también es un problema de seguridad nuevo al que se enfrentan los responsables de los departamentos informáticos.

Debido a los problemas de seguridad asociados a la utilización de los sistemas de mensajería instantánea, han comenzado a desarrollarse ya hace unos años, herramientas de gestión de la mensajería instantánea. Estas herramientas tienen la finalidad de garantizar mayor seguridad a las empresas que decidieron incorporar a la mensajería instantánea como medio de comunicación entre empleados y con sus clientes. En este proyecto se realizó un estudio del estado del arte de dichas aplicaciones de gestión para la mensajería instantánea. En este estudio se analizaron las capacidades de filtrado, y se verificó que muchas aplicaciones de gestión tienen la posibilidad de filtrar contenido de las conversaciones de mensajería, utilizando listas de palabras no permitidas. Sin embargo no son capaces de detectar el tráfico generado por IPoIM, de manera que no evitan la utilización de la mensajería con otra finalidad que no sea la de establecer conversaciones entre personas. La posibilidad de detectar el tráfico generado por IPoIM se basa en la búsqueda de patrones. Por ejemplo, cuando se utiliza la codificación con el método Base64, se pueden detectar mensajes codificados utilizando dicho método ya que los mensajes presentan un patrón similar. Esta detección se puede lograr con algunas aplicaciones de gestión de mensajería instantánea como por ejemplo Symantec IM Manager que es capaz de detectar

tráfico utilizando expresiones regulares. Sin embargo si se utilizan métodos de codificación que utilicen palabras válidas, la detección del tráfico generado por IPoIM es difícil ya que el tráfico se compone justamente de palabras válidas de algún idioma escrito. Con este tipo de codificación no sirven de nada la detección y el filtrado mediante listas de palabras ya que difícilmente se bloqueen palabras totalmente válidas y que no significan ningún riesgo para la empresa. Si se filtraran palabras válidas, se estaría impidiendo que las conversaciones reales de mensajería puedan establecerse, al menos cuando se utilizan las palabras bloqueadas. Esto no tendría sentido ya que en ese caso seria más conveniente no permitir ningún tipo de tráfico de mensajería, impidiendo el uso de estas aplicaciones.

Debido a que uno de los objetivos del presente proyecto es justamente analizar las posibilidades de filtrado de las herramientas de gestión de mensajería se desarrollo nuevo método de codificación utilizando palabras válidas. El método con diccionario de palabras. Con este método se demostró que las herramientas de gestión no son capaces de detectar el tráfico de IPoIM cuando éste utiliza palabras válidas. Esto se debe a que un filtro de mensajería, en una aplicación de gestión, no se configuraría con palabras válidas ya que se impediría el uso de la mensajería instantánea como tal. Es por ello que se concluye que si se utilizan métodos de codificación más sofisticados, los filtros de mensajería no son muy efectivos. Sin embargo, en estos casos cobran importancia las estadísticas de mensajes intercambiados en conversaciones de mensajería. Con las estadísticas se puede determinar si una conversación es mantenida por dos personas o en otro caso, la misma es generada por una aplicación como puede ser IPoIM. Como se detalló en el capítulo 5 que hacía referencia a los métodos de codificación y filtrado, las personas pueden generar un flujo de datos por un canal de mensajería mucho menor a el flujo de datos generados por una aplicación. Con esto se determinó que las estadísticas son muy importantes para garantizar la seguridad de las aplicaciones de mensajería. Muchas aplicaciones de gestión recolectan estadísticas de mensajes, pero ninguna bloquea las conversaciones que excedan los valores normales de mensajes enviados por minuto, o el largo de los mensajes, etc.

Por otro lado el trabajo realizado con las aplicaciones de gestión de mensajería escogidas para realizar pruebas fue exitoso debido a que se modificó una aplicación existente (IMSpector), capaz de registrar conversaciones de mensajería, para que bloquee las conversaciones cuando detecta que los mensajes están codificados utilizando el método de codificación Base64. Del mismo modo, utilizando otras herramientas (Symantec IM Manager, IM SecurityNet) también se bloqueó el tráfico codificado en Base64 utilizando patrones. Paralelamente se desarrolló un método de detección de IPoIM basado en estadísticas de mensajes agregando también a la aplicación IMSpector, la posibilidad de recolectar estadísticas de mensajes, con lo cual se obtuvieron resultados interesantes en cuanto al flujo de mensajes instantáneos generados por IPoIM.

Por último este proyecto brinda información acerca de las buenas prácticas para el uso seguro de las aplicaciones de mensajería instantánea. Se detallan las acciones a tomar por parte de una empresa que decida implementar una solución de mensajería instantánea o que, se haya visto obligada a tomar acciones debido a que la proliferación de las aplicaciones de mensajería cada vez es mayor y existe actualmente una variada oferta de aplicaciones de este tipo.

6.2. Conclusión del trabajo.

La mensajería instantánea presenta problemas de seguridad que deben ser tratados adecuadamente. Se concluye que los sistemas de mensajería instantánea no son un medio seguro, a pesar de los intentos realizados por las empresas que comercializan soluciones corporativas para la gestión de estas aplicaciones potencialmente peligrosas. El uso correcto de la mensajería instantánea tiene impacto en la seguridad. Pero tiene más impacto aún el uso de la mensajería con otros fines como el demostrado en este proyecto. La posibilidad de establecer un túnel VPN sobre una comunicación de mensajería instantánea abre un abanico de posibilidades. Con ello se puede establecer un canal encubierto para acceder a redes privadas, representando una amenaza para las empresas.

Con la mensajería instantánea se obtienen muchos beneficios, pero si no se controla debidamente, los beneficios se pueden ver opacados por los problemas de seguridad que pueden ocasionar pérdidas muy grandes a las empresas. Las amenazas y vulnerabilidades que presentan los sistemas de mensajería instantánea no se pueden descuidar ya que además de existir muchos riesgos, cada vez más los sistemas de mensajería son el objetivo de ataques por parte de hackers. Los virus o gusanos ven en éstas aplicaciones una excelente posibilidad de propagarse a través de las redes en muy poco tiempo. Las aplicaciones de mensajería instantánea son una herramienta muy útil para las organizaciones, pero se deben incrementar las medidas de seguridad para asegurar la correcta utilización de las mismas, y al mismo tiempo proteger la integridad de la información de las empresas y de sus activos. Por su parte, las herramientas de gestión de mensajería son recomendables para lograr un mayor nivel de seguridad en las aplicaciones de mensajería, pero éstas herramientas no protegen totalmente a los sistemas de las amenazas existentes. Se tiene que avanzar más en este aspecto para controlar el tráfico no relacionado con la mensajería de manera segura para los usuarios.

6.3. Trabajo futuro.

Este proyecto reveló que establecer un túnel IP sobre la mensajería instantánea implica mantener conversaciones con tasas elevadas de mensajes enviados por minuto, así como longitudes de mensajes excesivos para conversaciones reales entre seres humanos. Con esto se podría identificar un uso indebido de las aplicaciones de mensajería y consecuentemente, el uso de IPoIM.

Por lo expresado anteriormente se podría investigar si es posible establecer el túnel IP, pero simulando una conversación real entre personas, basándose en la cantidad de mensajes enviados por minuto y el largo de los mismos. La idea sería investigar si aún es posible usar protocolos como TCP con los nuevos tiempos de intercambio de mensajes asociados a una conversación real entre personas.

7. Glosario

Adware	Un programa adware es cualquier programa que automáticamente ejecuta, muestra o baja publicidad al computador después de instalado el programa o mientras se está utilizando la aplicación.
AIM	AIM (America-On-Line Instant Messenger) es un programa de mensajería instantánea de America On Line denominada habitualmente Instant Messenger.
AOL	AOL LLC (anteriormente America Online, Inc), es un proveedor estadounidense de medios y servicios de acceso a Internet operada por Time Warner.
Applet	Un applet es un componente de una aplicación que corre en el contexto de otro programa, por ejemplo un navegador web.
ASCII	El código ASCII (acrónimo inglés de American Standard Code for Information Interchange) Conjunto de normas de codificación de caracteres mediante caracteres numéricos, de amplia utilización en informática y telecomunicaciones.
Back door	Backdoor (puerta trasera), es un método escondido por el cual se puede acceder a un sistema pasando por alto cualquier tipo de autenticación.
Contacto	En mensajería instantánea se le llama contacto (buddy) a una entrada en la lista de personas con las que se desea conversar.
Eavesdropping	Eavesdropping, termino inglés que traducido al español significa escuchar secretamente, se ha utilizado tradicionalmente en ámbitos relacionados con la seguridad, como escuchas telefónicas o para interceptar conversaciones de mensajería instantánea
Firewalls	Un firewall es un equipo utilizado en redes de computadoras para controlar las comunicaciones, permitiéndolas o prohibiéndolas
Hacker	El término Hacker es utilizado para referirse a un experto (Gurú) en varias o alguna rama técnica relacionada con las tecnologías de la información y las telecomunicaciones: programación, redes de computadoras, sistemas operativos, hardware de red/voz, etc.
HIPAA	HIPAA es la ley federal de 1996 que se conoce como Ley de "Portabilidad" y Responsabilidad del Seguro Médico. La meta fundamental de la ley era facilitar a las personas el mantener un seguro médico, proteger la confidencialidad y la seguridad de la información del cuidado médico y ayudar a la industria del cuidado de la salud a controlar los costos administrativos

ICQ	ICQ ("I seek you" o "te busco") es un servicio de mensajería instantánea y el primero de su tipo en ser ampliamente utilizado en Internet, mediante el cual es posible chatear y enviar mensajes instantáneos a otros usuarios conectados a la red de ICQ.
IDS	Un sistema de detección de intrusos (o IDS de sus siglas en inglés Intrusion Detection System) es un programa usado para detectar accesos desautorizados a un computador o a una red.
IM	La mensajería instantánea es conocida también en inglés como IM.
IP	Internet Protocol.
IPoIM	Internet Protocol over Instant Messaging.
IPS	Un Sistema de Prevención de Intrusos (IPS) es un dispositivo que ejerce el control de acceso en una red informática para proteger a los sistemas computacionales de ataques y abusos.
Malware	Malware (del inglés malicious software, también llamado badware o software malicioso) es un software que tiene como objetivo infiltrarse en o dañar un ordenador sin el conocimiento de su dueño.
Mensajería Instantánea	Los mensajeros instantáneos son un conjunto de programas que utilizan el protocolo TCP IP que sirven para enviar y recibir mensajes instantáneos con otros usuarios conectados a Internet u otras redes.
MIME	MIME (Multipurpose Internet Mail Extensions), son una serie de convenciones o especificaciones dirigidas a que se puedan intercambiar a través de Internet todo tipo de archivos (texto, audio, vídeo, etc.)
MSN	MSN (abreviación de Microsoft Networks) es una colección de servicios de Internet proporcionado por Microsoft.
P2P	Peer to Peer. Así se conocen las aplicaciones punto a punto como Kazaa, Emule, etc.
Password	Una contraseña o clave (en inglés password), es una forma de autenticación que utiliza información secreta para controlar el acceso hacia algún recurso
PDA	PDA, del inglés Personal Digital Assistant, (Ayudante personal digital) es un computador de mano.
Phishing	Phishing es un término informático que denomina un tipo de delito encuadrado dentro del ámbito de las estafas, y que se comete mediante el uso de un tipo de ingeniería social caracterizado por intentar adquirir información confidencial de forma fraudulenta.
Smart phone	Un smartphone (teléfono inteligente) es un teléfono móvil que ofrece características avanzadas, típicamente como una computadora personal.
Spyware	Los programas espías o spywares son aplicaciones que recopilan información sobre una persona u organización sin su conocimiento.

TCP	Transmision Control Protocol.
TI	Tecnologías de la información.
Trojan horse	Se denomina troyano (o caballo de Troya, traducción fiel del inglés Trojan horse aunque no tan utilizada) a un programa malicioso capaz de alojarse en computadoras y permitir el acceso a usuarios externos, a través de una red local o de Internet, con el fin de recabar información o controlar remotamente a la máquina anfitriona.
Usuario federado	Los usuarios federados son aquellos que pertenecen a otras organizaciones con las cuales se permite el uso de mensajería, estableciendo entre éstas relaciones de confianza (certificados digitales).
Virus	Un virus informático es un programa que se copia automáticamente y que tiene por objeto alterar el normal funcionamiento de la computadora, sin el permiso o el conocimiento del usuario.
Worms	Un worm (gusano) es un virus informático que a diferencia de un virus, no precisa alterar los archivos de programas, sino que reside en la memoria y se duplica a sí mismo
Yahoo	Yahoo! Inc. es una empresa global de medios con sede en Estados Unidos. Entre sus productos esta el Yahoo Messenger.

8. Referencias

[1] Jive Software - Openfire: An open source real time communication (RTC) server (2008) [http://www.jivesoftware.com/products/openfire Accedido por última vez: 21-04-2008]
[2] Sigaba - Sigaba Secure IM (2008) [http://www.sigaba.com/sec_products/secure_im/index.shtml Accedido por última vez: 21-04-2008]
[3] Blue Coat Systems. - Web content filtering (2008) [http://www.bluecoat.com/products/webfilter Accedido por última vez: 21-04-2008]
[4] Yahoo Inc. - Yahoo Messenger home page (2008) [http://messenger.yahoo.com/ Accedido por última vez: 21-04-2008]
[5] Rick Jones - NetPerf Home page (2007) [http://www.netperf.org/netperf Accedido por última vez: 21-04-2008]
[6] Free Software Foundation, Inc. - Multi-Threaded Applications (2007) [http://www.gnu.org/software/gnutls/manual/html_node/Multi_002dthreaded-applications.html Accedido por última vez: 21-04-2008]
[7] Free Software Foundation, Inc. - GNU Autoconf, Automake and Libtool (2007) [http://sourceware.org/autobook/ Accedido por última vez: 21-04-2008]
[8] Free Software Foundation, Inc. - GTK+ The GIMP Toolkit (2007) [http://www.gtk.org Accedido por última vez: 21-04-2008]
[9] James Yonan - An Open Source SSL VPN Solution (2007) [http://openvpn.net Accedido por última vez: 21-04-2008]
[10] Sean Egan - The home of Pidgin, Finch, and the libpurple IM client library (2007) [http://www.pidgin.im Accedido por última vez: 21-04-2008]
[11] Philipp Winter - The IM-Filter Projectpage (2007) [http://im-filter.sourceforge.net/ Accedido por última vez: 21-04-2008]
[12] Jeremie Miller - Jabberd 2.x project (2007) [http://jabberd.jabberstudio.org/ Accedido por última vez: 21-04-2008]
[13] Ipswitch Inc. - Ipswitch Instant Messaging (2007) [http://www.ipswitch.com/products/instant_messaging/index.asp Accedido por última vez: 21-04-2008]
[14] WiredRed Software - Secure Instant Messaging (2007) [http://www.wiredred.com/secure-messaging Accedido por última vez: 21-04-2008]
[15] Liquid Communication Systems, LLC - Effusia Business Messenger (2007) [http://www.liquidcs.com/business-messenger/EffusiaBusinessMessenger.php Accedido por última vez: 21-04-2008]
[16] Sun Microsystems - Sun Java System Instant Messaging (2007) [http://www.sun.com/software/products/instant_messaging/index.xml Accedido por última vez: 21-04-2008]
[17] St. Bernard's - iPrism IM Filter (2007) [http://www.stbernard.com/products/liveprism/im_filter.asp Accedido por última vez: 21-04-2008]
[18] SANS - SANS Top-20 2007 Security Risks (2007 Annual Update) (2007) [http://www.sans.org/top20/ Accedido por última vez: 21-04-2008]
[19] Imlogic - IMlogic Threat Center (2007) [http://imlogic.com/im_threat_center/index.asp Accedido por última vez: 21-04-2008]
[20] ComScore Inc. - ComScore - Measuring the digital world (2007) [http://www.comscore.com/ Accedido por última vez: 21-04-2008]
[21] Skype Technologies S.A - Skype official website (2007) [http://www.skype.com/intl/en/?sid=1010 Accedido por última vez: 21-04-2008]
[22] Google Inc. - Google talk home page (2007) [http://www.google.com/talk/ Accedido por última vez: 21-04-2008]
[23] AOL - Instant Messenger - AIM - Instant Message Your Online Buddies for Free - AIM (2007) [http://dashboard.aim.com/aim Accedido por última vez: 21-04-2008]

[24] ICQ Inc. - ICQ.com - community, people search and messaging service (2007) [http://www.icq.com/ Accedido por última vez: 21-04-2008]
[25] Microsoft Corp. - Windows Live Messenger (2007) [http://get.live.com/messenger/overview Accedido por última vez: 21-04-2008]
[26] Lawrence Manning - Imspector: Instant Messenger transparent proxy (2006) [http://www.imspector.org Accedido por última vez: 21-04-2008]
[27] Tools That Work - IM Security Net: Instant Message Content Filter Service (2006) [http://www.toolsthatwork.com/im_security_net.htm Accedido por última vez: 21-04-2008]
[28] Symantec Corp. - Symantec IM Manager: Secure instant messaging (IM) management and policy enforcement of public and enterprise IM for enterprise risk management (2006) [https://portal.asap.com/en-US/Documents/Symantec_im_manager81.pdf Accedido por última vez: 21-04-2008]
[29] FaceTime Communications, Inc. - FaceTime IMAuditor (2006) [http://www.facetime.com Accedido por última vez: 21-04-2008]
[30] Microsoft Corp. - 10 tips for using instant messaging for business – Microsoft Corp. (2006) [http://www.microsoft.com/smallbusiness/resources/technology/communications/10_tips_for_using_instant_messaging_for_business.mspx Accedido por última vez: 21-04-2008]
[31] Symantec Corp. - Best Practices for IM Archivig and Compliance (2006) [http://www.symantec.com Accedido por última vez: 21-04-2008]
[32] Microsoft Corp. - All about phishing (2006) [http://www.microsoft.com/latam/seguridad/hogar/spam/phishing.mspx Accedido por última vez: 21-04-2008]
[33] E. Corti and A. Sabiguero - Seguridad de la Mensajería Instantánea: desafíos de IPoIM (2006) [http://www.fing.edu.uy/~asabigue/publi/mvdtelcom-015.pdf Accedido por última vez: 21-04-2008]
[34] A. Sabiguero, P. Rodríguez and L. Rodríguez - IPoIM: Internet Protocol over Instant Messaging (2006) [http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/4086332/4043248/04086434.pdf?tp=&arnumber=4086434&isnumber=4043248 Accedido por última vez: 21-04-2008]
[35] Laura Rodríguez - IPoIM: Internet Protocol over Instant Messaging (2006) [http://www.fing.edu.uy/~asabigue/prgrado/2005ipoim/docum_ipoim.pdf Accedido por última vez: 21-04-2008]
[36] Cisco Systems - Blocking Instant Messaging and Peer-To-Peer File Sharing Applications (2005) [http://www.cisco.com/ Accedido por última vez: 21-04-2008]
[37] Hummingbird - Hummingbird Enterprise IM (2005) [http://www.hummingbird.com Accedido por última vez: 21-04-2008]
[38] Akonix Systems, Inc. - Akonix L7 CM5000 Appliance Instant IM Control (2005) [http://www.akonix.com Accedido por última vez: 21-04-2008]
[39] SurfControl Inc. - Enterprises Unprepared To Manage Instant Messaging Threats (2005) [http://www.govtech.com/gt/articles/93530 Accedido por última vez: 21-04-2008]
[40] Laura Rohde - networkworld.com - Reuters shuts down system to fight Kelvir IM worm (2005) [http://www.networkworld.com/news/2005/0415reuteshuts.html Accedido por última vez: 21-04-2008]
[41] Leavitt Communications - Instant messaging: a new target for hackers (2005) [http://www.leavcom.com/ieee_july05.htm Accedido por última vez: 21-04-2008]
[42] AOL - AOL's Third Annual Instant Messenger Trends Survey (2005) [http://www.aim.com/survey/#Overall%20Usage%20Stats1 Accedido por última vez: 21-04-2008]
[43] Cade Metz - eweek.com - The Next Big Virus Threat: Instant Messaging (2004) [http://www.eweek.com/article2/0,1759,1617049,00.asp Accedido por última vez: 21-04-2008]
[44] Neal Hindocha - Instant Insecurity: Security Issues of Instant Messaging (2003) [http://www.securityfocus.com/infocus/1657 Accedido por última vez: 21-04-2008]
[45] CERT® Coordination Center - Denial of Service Attacks (2001) [http://www.cert.org/tech_tips/denial_of_service.html Accedido por última vez: 21-04-2008]

[46] Robert Graham - Packet Sniffing FAQ (2000) [http://web.archive.org/web/20050221103207/http://www.robertgraham.com/pubs/sniffing-faq.html Accedido por última vez: 21-04-2008]
[47] C. Perkins - RFC2003: IP Encapsulation within IP (1996) [http://www.ietf.org/rfc/rfc2003.txt Accedido por última vez: 21-04-2008]

9. Apéndice 1 – Implementación del plugin de filtrado para base64

```
/* IMSpector - Instant Messenger Transparent Proxy Service
 * http://www.imspector.org/
 * (c) Lawrence Manning <lawrence@aslak.net>, 2006
 *
 * Contributions from:
 *     Ryan Wagoner <ryan@wgnrs.dynu.com>, 2006
 *
 * Released under the GPL v2. */

//
// C++ Implementation: base64contentplugin
//
// Description:
//
//
// Author: Sebastian Nuñez <nunezs@gmail.com>, (C) 2007
//
// Copyright: See COPYING file that comes with this distribution
//
//

#include "imspector.h"

#define PLUGIN_NAME "Base64 IMSpector content plugin"
#define PLUGIN_SHORT_NAME "Base64"



extern "C"
{
      bool initcontentplugin(struct contentplugininfo
&pcontentplugininfo,
            class Options &options, bool debugmode);
      int contentfilter(char *originalbuffer, char *modifiedbuffer);
};



bool localdebugmode = false;

bool initcontentplugin(struct contentplugininfo &contentplugininfo,
      class Options &options, bool debugmode)
{
      if (options["enable_base64"] == "") return false;
      localdebugmode = debugmode;
      contentplugininfo.pluginname = PLUGIN_NAME;

      return true;
}

void closecontentplugin(void)
{
```

```
      return;
}

/* is_b64_chars BUF LEN
 * Are the first LEN bytes of BUF composed only of base64 characters?
*/
int is_b64_chars(const unsigned char *buf, size_t len) {
    const unsigned char *p;

    while (len > 0 && buf[len - 1] == '=')
        --len;
    for (p = buf+1; p < buf + len; ++p)
        if (!((*p >= 'A' && *p <= 'Z')
                || (*p >= 'a' && *p <= 'z')
                || (*p >= '0' && *p <= '9')
                || *p == '+' || *p == '/'))
            return 0;
    return 1;
}


/* The main plugin function. See contentplugin.cpp. */
int contentfilter(char *originalbuffer, char *modifiedbuffer)
{
      debugprint(localdebugmode, PLUGIN_SHORT_NAME ": filtering
before: original: %s",originalbuffer);
      if (is_b64_chars((const unsigned char *)originalbuffer,30)){
            char * s = modifiedbuffer;
            strcpy(modifiedbuffer,"Se detecto codificacion Base64");
            s=s+30;
            memset(s,(int) '*',strlen(originalbuffer)-30);
            }
      else strcpy(modifiedbuffer, originalbuffer);

      debugprint(localdebugmode, PLUGIN_SHORT_NAME ": filtering after:
modified: %s", modifiedbuffer);

      return 0;
}

void spaces(char* src, char* dst){

}
```

10. Apéndice 2 – Implementación de las estadísticas de mensajes

Archivo stats.h

```
#ifndef _STATS_H_
#define _STATS_H_

#include <stdio.h>
#include <stdlib.h>
#include <time.h>


typedef struct {
      char from[MAX_LENGTH];
      char to[MAX_LENGTH];
      int nMSGS;
      float avg;
      float len_avg;
      time_t t_start;
} _stats;

typedef struct {
      int tope;
      _stats stats[MAX_STATS];
} stats;


_stats* find(stats* st, char* from, char* to);
int create_stats(stats* st);
int write_stats_to_file(stats* st, char* file_name);
int conv_stats_create(stats* st, char* from, char* to);
int set_count_msg(stats* st, char* from, char* to, int count);
int set_avg_msg(stats* st, char* from, char* to, float avg);
int set_len_avg_msg(stats* st, char* from, char* to, float len_avg);

int get_count_msg(stats* st, char* from, char* to);
float get_avg_msg(stats* st, char* from, char* to);
float get_len_avg_msg(stats* st, char* from, char* to);

time_t get_time_init_conv(stats* st, char* from, char* to);
#endif
```

Archivo stats.cpp

```
#include "stats.h"
#include <search.h>
#include <string.h>
```

```
_stats* find(stats* st, char* from, char* to){
	int esta=0;
	int i=1;
	int pos;
	while ((i<=st->tope) && (!esta)){
		if((strcmp(from,st->stats[i].from)==0) && (strcmp(to,st-
>stats[i].to)==0)){
			esta=1;
			pos=i;
		}
		i++;
	}
	if (esta)
		return &(st->stats[pos]);
	else return NULL;
}

int create_stats(stats* st){
	st->tope=0;
	return 1;
}
int write_stats_to_file(stats* st, char* file_name){
	FILE* f;
	int i;
	printf("Escribiendo estadisticas...\n");
	if((f=fopen(file_name, "a"))==NULL)
		printf("Error: No se pudo abrir el archivo de estadisticas
%s\n",file_name);
	else{
		fprintf(f,"from:\tto:\tnumero de\tprom mensajes:\tlargo
promedio de mensaje:\n");
		fprintf(f,"\t\tmensajes:\t(mens/minuto)\t(cant.
caracteres)\n\n");
		for(i=1;i<=st->tope;i++){
			fprintf(f,"%s\t%s\t%d\t%4.2f\t%4.2f\n", st-
>stats[i].from, st->stats[i].to, st->stats[i].nMSGS, st->stats[i].avg,
st->stats[i].len_avg);
			printf("%s\t%s\t%d\t%4.2f\t%4.2f\n", st-
>stats[i].from, st->stats[i].to, st->stats[i].nMSGS, st->stats[i].avg,
st->stats[i].len_avg);
		}
	}
	fprintf(f,"\n\n");
	fclose(f);
	printf("Finalizado. Consulte las estadisticas en:
%s\n",file_name);
	return 1;
}
int conv_stats_create(stats* st, char* from, char* to){

	st->tope++;
	strcpy(st->stats[st->tope].from, from);
	strcpy(st->stats[st->tope].to, to);
	st->stats[st->tope].nMSGS=0;
	st->stats[st->tope].avg=0;
	st->stats[st->tope].len_avg=0;
	time(&(st->stats[st->tope].t_start));
	return 1;
```

```
}
int set_count_msg(stats* st, char* from, char* to, int count){
      _stats* p;
      p=find(st,from,to);
      if(p==NULL)
            printf("Error: No se encuentra la conversacion entre %s y
%s!\n",from,to);
      else
            p->nMSGS=count;
      return 1;
}
int set_avg_msg(stats* st, char* from, char* to, float avg){
      _stats* p;
      p=find(st,from,to);
      if(p==NULL)
            printf("Error: No se encuentra la conversacion entre %s y
%s!\n",from,to);
      else
            p->avg=avg;
      return 1;
}
int set_len_avg_msg(stats* st, char* from, char* to, float len_avg){
      _stats* p;
      p=find(st,from,to);
      if(p==NULL)
            printf("Error: No se encuentra la conversacion entre %s y
%s!\n",from,to);
      else
            p->len_avg=len_avg;
      return 1;
}


int get_count_msg(stats* st, char* from, char* to){
      _stats* p;
      p=find(st,from,to);
      if(p==NULL){
            printf("Error: No se encuentra la conversacion entre %s y
%s!\n",from,to);
            return 0;
            }
      else
            return p->nMSGS;
}

float get_avg_msg(stats* st, char* from, char* to){
      _stats* p;
      p=find(st,from,to);
      if(p==NULL){
            printf("Error: No se encuentra la conversacion entre %s y
%s!\n",from,to);
            return 0;
            }
      else
            return p->avg;
}

float get_len_avg_msg(stats* st, char* from, char* to){
```

```
	_stats* p;
	p=find(st,from,to);
	if(p==NULL){
		printf("Error: No se encuentra la conversacion entre %s y
%s!\n",from,to);
		return 0;
		}
	else
		return p->len_avg;
}

time_t get_time_init_conv(stats* st, char* from, char* to){
	_stats* p;
	p=find(st,from,to);
	if(p==NULL){
		printf("Error: No se encuentra la conversacion entre %s y
%s!\n",from,to);
		return 0;
		}
	else
		return p->t_start;
}
```

Modificaciones al archivo main.cpp

Las modificaciones realizadas a la aplicación IMSpector en su archivo principal se detallan a continuación:

```
	int shm_id;			/* ID  del segmento de memoria
compartida */
	void* shm_addr;		/* direccion del segmento de memoria
compartida */
	struct shmid_ds shm_desc;

	union semun {
		int val;
		struct semid_ds *buf;
		unsigned short  *array;
	} sem_val;		/* valor del semaforo */

	stats* st; /* estructura para almacenar las estadísticas */

	/* Crea el semaforo   */
	sem_set_id = semget(250, 1, IPC_CREAT | 0600);
	if (sem_set_id == -1) {
		perror("main: semget");
		exit(1);
	}

	/* Inicializa el semaforo a '1'. */
	sem_val.val = 1;
	r = semctl(sem_set_id, 0, SETVAL, sem_val);
```

```
      if (r == -1) {
            perror("main: semctl");
            exit(1);
      }

      /* Reserva el espacio de memoria compartida para albergar la
estructura de estadísticas. */
      shm_id = shmget(100, sizeof(stats), IPC_CREAT | IPC_EXCL |
0600);
      if (shm_id == -1) {
            perror("main: shmget: ");
            exit(1);
      }
      printf("shm_id: %d\n",shm_id);
      /* mapea el espacio de memoria compartida con el espacio de
memoria del proceso. */
      shm_addr = shmat(shm_id, NULL, 0);
      if (!shm_addr) {
            perror("main: shmat: ");
            exit(1);
      }


      /* Se asocia el segmento de memoria compartida con el puntero a
la esctructura de estadísticas */
      st = (stats*) ((void*)shm_addr);

      ....

      doproxy(options, debugmode, clientsock, clientaddress);

      ....

}
```

Entre otras cosas, en el archivo main se llama a la funcion doproxy() que es invocada para cada conversación establecida. El proceso doproxy() se ejecuta como subproceso del procesos principal. Por ello se necesitó la estructura de memoria compartida. El proceso doproxy() tambien debe mapear en su espacio de memoria a al segmento de memoria compartida que almacena a la estructura de estadísticas. De esta manera puede modificar los datos estadísticos para la conversación establecida.

Modificaciones al proceso doproxy()

```
bool doproxy(class Options &options, bool debugmode, class Socket
&clientsock, std::string &clientaddress)
{

      stats* st;
      int shm_id;                     /* ID  del segmento de memoria
compartida */
      void* shm_addr;          /* direccion del segmento de memoria
compartida */
      struct shmid_ds shm_desc;
```

```
      /* Localiza el segmento de memoria compartida. */
      shm_id = shmget(100, sizeof(stats), 0600);
      if (shm_id == -1) {
            perror("main: shmget: ");
            exit(1);
      }
      printf("shm_id: %d\n",shm_id);
      /* mapea el espacio de memoria compartida con el espacio de
memoria del proceso. */
      shm_addr = shmat(shm_id, NULL, 0);
      if (!shm_addr) {
            perror("main: shmat: ");
            exit(1);
      }

      /* Se asocia el segmento de memoria compartida con el puntero a
la esctructura de estadísticas */
      st = (stats*) ((void*)shm_addr);


      .....


      logimevents(imevents, clientaddress, st)


      .....
}
```

En determinado momento, y con la llegada de nuevos mensajes de mensajería se invoca a la función logimevents(), la que se encarga de realizar los calculos estadísticos (además de la función original de registro de mensajes).

Modificaciones al proceso logimevents()

```
bool logimevents(std::vector<struct imevent> &imevents, std::string
&clientaddress, stats* st)
{

            .....

            sem_lock(sem_set_id);

            if(find(st, (char*)(*i).localid.c_str(),
(char*)(*i).remoteid.c_str())==NULL){
                  conv_stats_create(st, (char*)(*i).localid.c_str(),
(char*)(*i).remoteid.c_str());
            }

            time(&t);
            /* Se calcula la cantidad de minutos transcurridos desde el
inicio de la conversación */
            mins = ((int)t-(int)get_time_init_conv(st,
(char*)(*i).localid.c_str(), (char*)(*i).remoteid.c_str()))/60;
```

```
            /* Se obtiene el largo del mensaje */
            largoMens = strlen((*i).eventdata.c_str());

            /* se cuenta la cantidad de mensajes hasta el momento */
            cantMSG = get_count_msg(st, (char*)(*i).localid.c_str(),
(char*)(*i).remoteid.c_str());

            /* Se calcula el promedio de mensajes por minuto */
            promMSG = get_avg_msg(st, (char*)(*i).localid.c_str(),
(char*)(*i).remoteid.c_str());

            /* se calcula el largo promedio de los mensajes */
            larPromMSG = get_len_avg_msg(st,
(char*)(*i).localid.c_str(), (char*)(*i).remoteid.c_str());
            larPromMSG = (largoMens + larPromMSG*cantMSG)/(cantMSG+1);
            set_len_avg_msg(st, (char*)(*i).localid.c_str(),
(char*)(*i).remoteid.c_str(), larPromMSG);

            cantMSG++;
            set_count_msg(st, (char*)(*i).localid.c_str(),
(char*)(*i).remoteid.c_str(), cantMSG);

            if(mins==0)
                  set_avg_msg(st, (char*)(*i).localid.c_str(),
(char*)(*i).remoteid.c_str(), cantMSG);
            else set_avg_msg(st, (char*)(*i).localid.c_str(),
(char*)(*i).remoteid.c_str(), (float)cantMSG/mins);

            sem_unlock(sem_set_id);


      .....

}
```

Como se describe en el código las estadísticas son recalculadas con la llegada de cada mensaje instantáneo, por lo que siempre estan actualizadas.

Cuando el programa (IMSpector) finaliza la ejecución, las estadísticas son escritas en un archivo de texto ubicado en /usr/etc/imspector/estadisticas.txt.

11. Apéndice 3 – Compilación, instalación y ejecución de IPoIM.

Dependencias

Para la correcta compilación y ejecución del prototipo se requieren ciertas dependencias de software que se detallan a continuación:

Sistema operativo Linux

Para la compilación de OpenVPN:

Se requiere:

(1) TUN / TAP driver para permitir a un programa ejecutándose en el espacio del usuario controlar dispositivos virtuales IP punto a punto, o dispositivos ethernet.

Opcional (pero recomendado):

(1) OpenSSL library, necesaria para encriptación. Se requiere version 0.9.5 o superior, disponible en http://www.openssl.org/

(2) LZO real-time compression library, requerida para la compresión del link. Disponible en http://www.oberhumer.com/opensource/lzo/

(3) Pthread library.

Opcional (Para desarrolladores):

(1) Autoconf 2.50 o superior + Automake 1.5 o superior.

-- Disponible en http://www.gnu.org/software/software.html

(2) Dmalloc library

-- Disponible en http://dmalloc.com/

Para la compilación de Pidgin:

(1) GTK+ Se requiere versión 2.6.10 o superior. (http://www.gtk.org/)

(2) GtkSpell / Aspell. Descargar los paquetes GtkSpell y Aspell.

(3) Mozilla NSS. Descargar Network Security Services (NSS) y Netscape Portable Runtime (NSPR) o GnuTLS

Sistema operativo Windows

Para la compilación de OpenVPN:

(1) MinGW/MSYS environment http://mingw.sourceforge.net/

(2) OpenSSL library http://www.openssl.org/

(3) LZO library http://www.oberhumer.com/opensource/lzo/

Para la compilación de Pidgin:

1- Instalar Cygwin Bash shell. (http://cygwin.com/) Asegurarse que se disponga de las siguientes utilidades: bash, bzip2, coreutils, gawk, grep, gzip, make, patch, sed, monotone, tar, unzip, and wget.

2- Instalar MinGW "current" packages desde el sitio de MinGW. http://www.mingw.org/ Específicamente se utilizarán gcc-core 3.4.2, binutils 2.15.91, mingw-runtime 3.9 y win32api 3.9 (o superior). Adicionalmente se necesita setear en la variable de entorno PATH el directorio bin de MinGW antes que el directorio de Cygwin. Ejemplo: export PATH=/cygdrive/c/MinGW/bin:$PATH.

3- Extraer los fuentes de pidgin (http://www.pidgin.im) en el directorio $PIDGIN_DEV_ROOT/pidgin-<version>, y luego instalar las siguientes dependencias en el directorio $PIDGIN_DEV_ROOT/win32-dev:

(4) GTK+ Se requiere versión 2.6.10 o superior. (http://www.gtk.org/) Extraer el archivo gtk-dev-2.6.10-rev-a.tar.gz en el directorio $PIDGIN_DEV_ROOT/win32-dev.

(5) Libxml. Descarcar y extraer en el directorio win32-dev el archivo libxml2-2.6.30.tar.gz.

(6) Perl 5.8. Instalar Perl .8 para windows en c:\Perl. (http://www.activestate.com/) Tambien es necesario instalar el archivo perl582.tar.gz en el directorio $PIDGIN_DEV_ROOT/win32-dev.

(7) Tcl 8.4.5. Descargar y extraer el archivo tcl-8.4.5.tar.gz en el directorio win32-dev.

(8) GtkSpell / Aspell. Descargar los paquetes GtkSpell (gtkspell-2.0.11-daa1.tar.gz) y Aspell (aspell-dev-0-50-3-3.zip), y extraerloes en el directorio win32-dev.

(9) Mozilla NSS. Descargar y extraer en el directorio $PIDGIN_DEV_ROOT/win32-dev los siguientes: Network Security Services (NSS) y Netscape Portable Runtime (NSPR)

(10) SILC Toolkit. Descargar y extraer el archivo silc-toolkit-1.1.2.tar.gz en el directorio win32-dev.

(11) Meanwhile. Descargar y extraer en el directorio $PIDGIN_DEV_ROOT/win32-dev el archivo meanwhile-1.0.2-win32.zip.

(12) Bonjour SDK. Descargar el SDK de Bonjour desde el website de Apple e instalarlo en el directorio win32-dev/Bonjour_SDK.

Por mas información consultar la página http://developer.pidgin.im/wiki/BuildingWinPidgin

Compilación de los fuentes.

Los pasos a seguir para instalar el prototipo son:

1- Descargar los fuentes de la aplicación OpenVPN, o utilizar la copia provista en el cd entregado junto con la documentación. Se utilizó la versión 2.0.9. (http://openvpn.net)

2- Descarcar los fuentes de la aplicación Pidgin, o utilizar la copia provista en el cd entregado junto con la documentación. Se utilizó la version 2.1.0. (http://www.pidgin.im)

3- Descomprimir el archivo ipoim-2.0.tar.gz (proporcionado en el cd adjunto en el directorio /src) en el directorio $(INST_DIR). Se generará el directorio ipoim-2.0.

```
prompt#>    cd $(INST_DIR)
prompt#>    tar -zxvf $(CDROM)/src/ipoim-2.0.tar.gz
```

4- Moverse al directorio ipoim-2.0 y descomprimir en él los fuentes de openvpn-2.0.9.tar.gz.

```
prompt#>    cd $(INST_DIR)/ipoim-2.0
prompt#>    tar -zxvf $(CDROM)/src/openvpn-2.0.9.tar.gz
```

5- Del mismo modo en el directorio ipoim-2.0 descomprimir los fuentes de pidgin-2.1.0.tar.gz.

```
prompt#>    tar -zxvf $(CDROM)/src/pidgin-2.1.0.tar.gz
```

6- Moverse al directorio openvpn-2.0.9 Y Aplicar el parche parche.openvpn.diff al mismo.

```
prompt#>    cd $(INST_DIR)/ipoim-2.0/openvpn-2.0.9
prompt#>    patch -p1 < $(CDROM)/src/parche.openvpn.diff
```

7- Moverse al directorio pidgin-2.1.0 y aplicar el parche parche.pidgin.diff al mismo.

```
prompt#>    cd $(INST_DIR)/ipoim-2.0/pidgin-2.1.0
prompt#>    patch -p1 < $(CDROM)/src/parche.pidgin.diff
```

8- Moverse al directorio ipoim-2.0 y ejecutar el script `configure`.

```
prompt#>    cd $(INST_DIR)/ipoim-2.0
prompt#>    ./configure --disable-consoleui --disable-screensaver
--disable-startup-notification --disable-gtkspell --disable-cap
--disable-schemas-install --disable-gstreamer --disable-fortify
--disable-tcl --disable-tk --disable-doxygen --disable-sm --
disable-largefile --disable-perl --enable-dbus=no --enable-nm=no
--enable-mono=no --enable-dot=no
```

NOTA: Las opciones pasadas al script configure conforman el conjunto de opciones para las cuales el prototipo esta desarrollado. Con estas opciones se deshabilitan las características que no van a ser utilizadas por IPoIM.

9- Compilar el prototipo

```
prompt#>    make
```

10- Instalar el prototipo

```
prompt#>    make install
```

Una vez finalizado el proceso se puede encontrar el ejecutable **ipoim** en el directorio $(INST_DIR)/ipoim-2.0/openvpn-2.0.9.

Ejecución

Para ejecutar el prototipo es necesario primero crear un archivo de clave estática.

```
prompt#>    ipoim --genkey --secret static.key
```

Luego es necesario copiar dicho archivo en ambos extremos del túnel, es decir, cada instancia de ipoim debe utilizar el mismo archivo de clave estática. El archivo debe ser copiado por un medio seguro.

Por otra parte cada instancia de ipoim debe utilizar un archivo de configuración donde se especifican ciertos parámetros.

Para la instancia que tiene el rol de servidor un ejemplo de archivo de configuración es el siguiente:

server.conf:

```
local 127.0.0.1 1194
dev tun
ifconfig 192.168.0.1 192.168.0.2
secret static.key
ipoim asy-ipoim@fing.edu.uy prpl-msn mlr-ipoim@fing.edu.uy Base64
tun-mtu 1500
```

Las opciones especificadas en el archivo se detallan a continuación:

`dev tun:`
Le decimos a la aplicación que vamos a usar un dispositivo de tipo tun para utilizar una conexión punto a punto.

`ifconfig 192.168.0.1 192.168.0.2:`
Especificamos las direcciones IP de ambos extremos de la conexión.

`secret static.key:`
Con este parámentro le decimos que vamos a utilizar el archivo "static.key" como clave estática en nuestra conexión.

`ipoim asy-ipoim@fing.edu.uy prpl-msn mlr-ipoim@fing.edu.uy Base64:`
Aquí configuramos las cuentas de mensajería que vamos a utilizar y el protocolo. La primera es utilizada por el servidor y la segunda es utilizada por el cliente. Al mismo tiempo se específica el método de codificación que se va a utilizar. En este caso "Base64"

`tun-mtu 1500:`
Este parámetro especifica el tamaño de mtu a utilizar en la conexión.

Por otro lado, para la instancia de ipoim con rol de cliente un archivo de configuración sería:

client.conf:

```
remote 127.0.0.1 1195
local 127.0.0.1 1194
dev tun
ifconfig 192.168.0.2 192.168.0.1
secret static.key
ipoim mlr-ipoim@fing.edu.uy prpl-msn asy-ipoim@fing.edu.uy Base64
tun-mtu 1500
```

Los parámetros en el cliente son similares a los del servidor. La diferencia es el orden de las direcciones IP y de las cuentas de mensajería.

Cuentas de mensajería:

Las cuentas de mensajería que son pasadas como parámetros en los archivos de configuración antes descriptos deben darse de alta en un archivo en los PCs que participan del túnel IP. IPoIM busca las cuentas de mensajería en el archivo llamado *accounts.xml* ubicado en el directorio del usuario, en la carpeta *.pidgin*. Existen dos métodos para realizar esto.

1- Editar (o crear si no existe) el archivo *accounts.xml* en el directorio *$(USER_DIR)/.purple/* con la información de las cuentas de mensajería.
2- Usar la interfaz gráfica de Pidgin para dar de alta las cuentas.

Para la opción 1 el procedimiento es el siguiente:

Editar o crear el archivo *$(USER_DIR)/.purple/accounts.xml* con la siguiente estructura para cada cuenta que se desea agregar:

```
<?xml version='1.0' encoding='UTF-8' ?>

<accounts version='1.0'>

 <account>
  <protocol>PROTOCOL</protocol>
  <name>ACCOUNT</name>
  <password>PASSWORD</password>
  <settings>
   <setting name='check-mail' type='bool'>0</setting>
   <setting name='server' type='string'>SERVER</setting>
   <setting name='http_method' type='bool'>0</setting>
   <setting name='port' type='int'>SERVER_PORT</setting>
  </settings>
  <settings ui='gtk-gaim'>
   <setting name='auto-login' type='bool'>0</setting>
  </settings>
  <proxy>
   <type>none</type>
  </proxy>
 </account>

</accounts>
```

Por ejemplo, para la cuenta de MS Messenger asy-ipoim@fing.edu.uy con password *clave123*, se debe editar el archivo con la siguiente información:

```
 <account>
  <protocol>prpl-msn</protocol>
  <name>asy-ipoim@fing.edu.uy</name>
  <password>clave123</password>
  <settings>
   <setting name='check-mail' type='bool'>0</setting>
   <setting name='server' type='string'>messenger.hotmail.com</setting>
   <setting name='http_method' type='bool'>0</setting>
   <setting name='port' type='int'>1863</setting>
  </settings>
  <settings ui='gtk-gaim'>
```

```
  <setting name='auto-login' type='bool'>0</setting>
 </settings>
 <proxy>
  <type>none</type>
 </proxy>
</account>
```

Para la opción 2 el procedimiento es el siguiente:

Ejecutar la aplicación Pidgin, ir a la ventana de alta y/o modificación de cuentas de mensajería que se muestra en la figura 16, y completar los datos solicitados en el formulario.

Figura 16: Pidgin - Ingreso de cuenta de mensajería

Finalmente, luego de ingresar las cuentas de mensajería, y de crear los archivos de configuración, para ejecutar el prototipo se debe realizar de la siguiente manera:

```
prompt#> ipoim server.conf
```

Para levantar un servidor

```
prompt#> ipoim client.conf
```

Para levantar un cliente

En ambos casos el archivo *static.key* debe estar en el directorio actual.