Creación de Patrones de Criptografía PGP Para Aplicaciones Utilizando Linux

M. en C. Jesús Antonio Álvarez Cedillo M. en C. Elizabeth Acosta Gonzaga Ing. Macario García Arregui

Profesores del CIDETEC-IPN

a seguridad informática comprende muchas áreas de interés originadas por la importancia fundamental del valor de la información. El garantizar que la información que es transmitida por cualquier medio de comunicación sea inaccesible para otras personas, no es un proceso fácil, y en algunas ocasiones requiere de recursos computacionales y financieros considerables; existen muchos esquemas de encriptación, sistemas y algoritmos para proteger la información mientras ésta se transmite, de tal manera que sea accesible unicamente por quien tiene la autorización de leerla.

El sistema operativo *Linux* ha servido como una plataforma de arranque importante para el desarrollo de nuevos estándares y sistemas abiertos, donde la colaboración de muchas personas expertas en software permite la creación de protocolos y sistemas nuevos.

El Sistema PGP (Pretty Good Privacy ó Muy Buena Privacidad)[1] es un sistema de encriptación por llave pública utilizado en plataformas Linux que actualmente se está desarrollando como un estándar, que utiliza dos llaves una pública y otra privada; la llave pública es la que se distribuye a los usuarios autorizados, y sirve para el envio de mensajes codificados que solo pueden decifrarse mediante la llave privada. Esta llave pública también puede servir para firmar un mensaje poniendo una parte de la llave privada en la firma, considerándose esto como un certificado de autenticidad; así, al recibir el mensaje el PGP comprueba la firma y texto y lo compara con la llave pública que el destinatario recibe previamente del remitente, mostrando un error si se ha cambiado algo en el texto o la rúbrica electrónica no corresponde a la de la persona que envía el mensaje. Su versión en software libre, GnuPG, es una utilidad de línea de comandos que incluye al motor de cifrado, mismo que puede ser utilizado directamente desde dicha línea, desde programas de intérpretes de instrucciones o por otros programas, siendo esta la característica que permite crear servidores de seguridad.

HISTORIA

Las ideas de la innovación del sistema *PGP* eran comunes y generales entre los informáticos y los matemáticos desde hace mucho tiempo, por lo que sus conceptos subyacentes no son en verdad innovadores.

Philip R. Zimmermann es el creador del *Pretty Good Privacy*, para la encriptación de correo electrónico; PGP se publico gratis en la Internet en 1991, y se convirtió en el software de encriptación más utilizado en el mundo. Zimermann es asesor en cuestiones de criptografía para varias compañías y organizaciones industriales, además es socio del Stanford Law School's Center for Internet and Society (Centro de Enseñanza de Derecho de Stanford para Internet y la Sociedad).

La innovación verdadera de Zimmermann consistía en la creación de estas herramientas para ser utilizadas por cualquier persona con una computadora personal; incluso las primeras versiones del *PGP* dieron acceso a las personas que utilizaban el sistema operativo *MSDOS*. Zimmermann, activista antinuclear, pensó que el *PGP* sería empleado por la mayoría de los disidentes y de los rebeldes; es decir este sistema se usaría fuera y dentro de los Estados Unidos de Norteamérica para el envío de información confidencial activista.

Desde el fin de la segunda guerra mundial, el gobierno de Estados Unidos ha considerado la encriptación resistente como una amenaza seria para la seguridad nacional, por lo que prohíbe exportar este sistema de los Estados Unidos al resto del mundo; la exportación de software del cifrado, incluyendo el PGP, requirió una licencia del Departamento

XVII 1 34

Directorio	de C:∖gn	սքց		
02/05/2006	13:47	<dir></dir>		-
02/05/2006	13:47	<dir></dir>		
16/12/1999	10:17		77.508	de.mo
07/04/1999	22:05		65.536	entropy.dll
16/12/1999	10:17		70.446	es_ES.mo
16/12/1999	10:17		74.321	fr.mo
19/12/1999	15:40		468.480	gpg.exe
19/12/1999	15:40		40.925	gpg.man
16/12/1999	10:17		68.582	id.mo
16/12/1999	10:17		72.358	it.mo
16/12/1999	10:17		73.772	pl.mo
16/12/1999	10:17		73.014	pt_BR.mo
16/12/1999	10:17		72.751	pt_PT.mo
02/05/2006	13:47		0	pubring.gpg
04/12/1999	14:32		20.303	README
06/12/1999	08:45		3.850	README.W32
16/12/1999	10:17		25.339	ru.mo
02/05/2006	13:47		0	secring.gpg
0000	16 arc]	hivos	1.207	7.185 bytes
2 dirs 45.749.792.768 bytes libres				

pgp263ii.zip y un archivo de llave publica denominado pgp263ii.asc. Ambos archivos contienen la información para verificar la llave pública de Stale Schumacher que indica si el archivo es auténtico. La llave de Stale resulta al descifrar el archivo keys.asc, que también es resultante de la descompresión del archivo antes mencionado (pgp263i.zip). En la **Figura 1** se muestra la estructura de archivos creados.

FUNCIONAMIENTO

Para generar el par de llaves, tanto la publica como la privada, es

del Estado, quedando vedado tanto el software, como la licencia de uso para ciertos países, los cuales no podrían recibir tales exportaciones bajo circunstancia alguna. Estas reglas eran conocidas como *ITAR*, quedando clasificadas las herramientas de cifrado como armas de guerra en el área de las comunicaciones en el ejército de los Estados Unidos de Norteamérica.

El gobierno, con este fundamento, demandó a Zimmermann quien por tres años defendió su proyecto en las cortes. Este pleito dio vuelta al mundo, y Zimmermann fue considerado como un héroe en la comunidad computacional, por lo que mucha gente descargaba el *PGP* para ver cuál era el problema y en que consistía el reclamo hecho por el gobierno, causando así un boom informático. La defensa de Zimmermann separó las noticias del pleito del PGP y las presentó en las audiencias, donde leyó las cartas que había recibido de la gente proveniente de los regímenes opresivos y de las áreas de devastación de guerra, cuyas vidas habían sido salvadas por *PGP*.

Сомо obtener PGP

El archivo de instalación del PGP se llama pgp263i. zip y es posible obtenerlo por Internet; el programa, información y auxiliares se encuentran en **http://www. pgpi.com/**, la cual es la página internacional del PGP. El archivo deberá descomprimirse con PKUNZIP, de modo que sus componentes queden guardados en el directorio c:\pgp, o bien c:/home/usuario/pgp; aparecerán algunos archivos, entre ellos otro elemento comprimido llamado necesario que el programa esté debidamente cargado y configurado. Es importante que esté el nombre del usuario en el config.txt y que se escriba de la misma forma en el par de llaves. El comando para generar dicho par es muy sencillo (**Figura 2**).

gpg –gen –key

Luego el programa preguntará por el tamaño de la llave, (ver **Figura 3**). El nivel de seguridad está en relación con el tamaño de las claves (medido en bits) que se generarán al cifrar cada vez. Cuanto mayor sea el número de bits, los procesos serán más lentos. Con la velocidad de las máquinas actuales, 1024 bits es razonablemente rápido y con mucha seguridad.

Después preguntará por el tiempo de expiración de la llave, (ver **Figura 4**). Se recomienda seleccionar la opción en la que la llave nunca caduque y de esta forma el usuario no tendrá que preocuparse por la vigencia; a continuación será necesario identificar la llave con un usuario, esto se observa en la **Figura 5**.

Después solicitará la frase que irá integrada en la llave, formada por la frase clave o contraseña que servirá para abrir la llave privada. Se recomienda una frase no muy corta y que no se adivine fácilmente, que no sea una frase hecha y que de preferencia tenga algunos signos de puntuación, números o caracteres en código ASCII simple. Es necesario mover el teclado y el ratón con el fin de generar mayor carga computacional que se traduzca en números al azar, (ver **Figura 6**.)

Creación de Patrones de Criptografía PGP para Aplicaciones Utilizando Linux

C:\WINDOWS\system32\cmd.exe - gpg --gen-key - 🗆 🗙 ۰ C:\gnupg>gpg --gen-key gpg: Please note that you don't have secure memory on this system gpg (GnuPG) 1.0.1a; Copyright (C) 1999 Free Software Foundation, Inc. This program comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to redistribute it under certain conditions. See the file COPYING for details. gpg: NOTE: THIS IS A DEVELOPMENT VERSION! gpg: NOTE: Ints is H DECELORMENT DERSION; gpg: It is only intended for test purposes and should NOT be gpg: used in a production environment or with production keys? Please select what kind of key you want: (1) DSA and ElGamal (default) (2) DSA (sign only) (4) ElGamal (sign and encrypt) Your selection? _ siguiente: \mathbf{x} 4 F Figura 2. Uso de generación de llaves. Your selection? 1 DSA keypair will have 1024 bits. About to generate a new ELG-E keypair. minimum keysize is 768 bits dofault keysize is 1024 bits highest suggested keysize is 2048 bits What keysize do you want? (1024) de la siguiente orden: Figura 3. Muestra el tamaño por definir en bits. **O**PERACIÓN Please specify how long the key should be valid. 0 = key does not expire $\langle n \rangle$ = key expires in n days $\langle n \rangle w$ = key expires in n weeks Figura 4. Muestra el tiempo de expiración. ASCII y no un archivo binario, (**Figura 7**): Real name: antonio alvarez Email address: jaalvarez@ipn.mx Comment: antoine You selected this USER-ID: "antonio alvarez (antoine) (jaalvarez@ipn.mx)" Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? _ Figura 5. Muestra el nombre de usuario que conforma las llaves. We need to generate a lot of random bytes. It is a good idea to perform the gpg: C:/Documents and Settings/cidetec/Datos de programa/gnupg\trustdb.gpg: trus tdb created gpg: key 745D9654 marked as ultimately trusted public and secret key created and signed. gpg: checking the trustdb gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u pub 1024D/745D9654 2006-05-02 Key fingerprint = FF12 EE2E 6F22 CB6F 8420 18EA 9585 F29F 745D 9654 antonio alvarez (antoine) (jaalvarez@ipn.mx) 2048g/0699BF62 2006-05-02 uid sub asc Figura 6. Muestra el nombre que va a estar en las llaves.

Para poder intercambiar mensajes cifrados con otra persona, esta debe conocer nuestra llave pública y nosotros la suya. Cabe considerar que para que el usuario pueda crear su llave, utilizará la instrucción

\$gpg -export -a -u jesusantonioa@ipn. mx > pubkey.asc

Para importar la llave del usuario remoto con el cual se intercambiará información se debe utilizar la siguiente instrucción:

\$gpg -import usuario.pubkey.asc

Para validar la llave o darle el visto bueno se hará uso

\$gpg -edit-key usuario

Para la operación de este sistema, se debe usar el archivo de texto llamado tclaro.txt (Texto claro) que contiene la palabra CIDETEC. Primero deberá ser firmado electrónicamente empleando el comando SIGN y utilizando la opción -a para que el resultado sea un archivo de 7 bits

GPG -a -sign tclaro.txt

A continuación se hace la verificación del archivo utilizando el comando (**Figura 8**):

gpg –veryfy TCLARO.TXT.asc

Después sólo es necesario mandarlo al destinatario utilizando cualquier medio: Internet, correo electrónico, etc.

Debe observarse el contenido del archivo, considerándose que ahora es un archivo cifrado y obviamente para descifrarlo es necesario aplicar el siguiente comando (**Figura 9**):

Gpg -o tdescifrado.txt -decrypt TCLARO.TXT.

Creación de Patrones de Criptografía PGP para Aplicaciones Utilizando Linux

C:\ARCHIV~1\GNU\GnuPG>GPG -a --sign TCLARO.TXT

You need a passphrase to unlock the secret key for user: "antonio alvarez (antoine) (jaalvarez@ipn.mx)" 1024-bit DSA key, ID 745D9654, created 2006-05-02

Figura 7. Muestra la manera de convertir y firmar electrónicamente el archivo tclaro.txt (Para realizarlo es necesario la palabra clave).

C:\ARCHIV~1\GNU\GnuPG>gpg —-verify TCLARO.TXT.asc gpg: Signature made 05/03/06 13:24:39 using DSA key ID 745D9654 gpg: Good signature from "antonio alvarez (antoine) {jaalvarez@ipn.mx>"

Figura 8. Muestra la manera de verificar el archivo tclaro.txt.asc (Para realizarlo se necesita la palabra clave)

C:\Archivos de programa\GNU\GnuPG>more TCLARO.TXT.asc -----BEGIN PGP MESSAGE------Version: GnuPG v1.4.2 (MingW32)

owGbwMvMwCQ4tfXT/JLYaSGMaySTuEKcfRyD/PUCIkJcIr6m03u6uIa40vNyddgz szKARGCKBZnS9zHMT08/070qoWLbmzi++Q/d0CIkbyxUYZjNuvNk7gSJT5f+3Gs5 eeJØzOngu5V/AA= =BKo1

END PGP MESSAGE

Figura 9. Muestra el archivo encriptado

C:\ARCHIV~1\GNU\GnuPG>more dtexto.txt **CIDETEC** Figura 10. Muestra la manera de desencriptar el archivo tclaro.txt.asc

Lo anterior se muestra en la **Figura 10**.

Para comprobar el contenido, debe observarse la igualdad con el original, por lo que el proceso de operación se ha terminado, y lo más importante es que fue completamente seguro.

CONCLUSIONES

En el seguimiento de las instrucciones del proceso de encriptación y desencriptación usando esta herramienta, se observa que es sencillo, rápido y dinámico, ya que utilizando solo algunas órdenes de comando simples es posible integrarlo a procesos y aplicaciones complejos.

Por otro lado, el entorno de utilización también es simple, haciendo así posible el integrarlo de manera multiplataforma a proyectos y de manera de multiprogramación en el uso de diferentes entornos que soporten el GnuPG, empleando procesos secuénciales y paralelos

y permitiendo inclusive la construcción de servidores distribuidos y servicios autónomos de encriptación y desencriptacion vía conexión de una red de área local o bien a través de la Internet.

REFERENCIAS

Pagina Oficial Pagina sobre criptografía Pagina de PGP Internacional **www.pgpi.com** Phil Zimmermann

www.gnugp.org www.kriptopolis.com web.mit.edu/~prz

BIBLIOGRAFÍA

- [1] Samson Garpike. PGP: Pretty Good Privacy. Dec 1, 1994
- Michael Lucas. PGP & GPG: Email for the Prac-[2] tical Paranoid. April 1, 2006.