PROPOSTA E IMPLEMENTAÇÃO DE PROTOCOLO DE TRANSFERÊNCIA DE ARQUIVOS USANDO SEGURANÇA POR CHAVE PÚBLICA (Fast-TP)

Krishnan Lage Pontes
krishnan@inf.ufsc.br
Universidade Federal de Santa Catarina
Laboratório de Redes e Gerência

Carla Merkle Westphall
carla@lrg.ufsc.br
Universidade Federal de Santa Catarina
Laboratório de Redes e Gerência

Carlos Becker Westphall
westphal@lrg.ufsc.br
Universidade Federal de Santa Catarina
Laboratório de Redes e Gerência

Resumo

O Protocolo de transferência de arquivos (FTP) é muito bem definido, implementado e utilizado pelo mundo. Apesar disto, algumas situações ocorrem onde seria desejável ter uma forma mais simples, rápida, mas ainda confiável de transmitir informações entre cliente e servidor. Neste trabalho apresentamos uma situação real de um projeto de cartão convênio onde esta necessidade se faz presente. Estudamos o FTP e o Telnet e descrevemos implementação do Fast-TP (Fast Transmission Protocol). A seguir efetuamos testes comparativos entre o Fast-TP e o FTP tradicional. Finalmente apresentamos aspectos de segurança da implementação do Fast-TP.

Abstract

The File Transfer Protocol is very well implemented, tested and widely used. Despite of that there is some situations where it would be desirable to have more simple, faster, but still reliable way of transmiting information between client and server. At this paper we present a real life Card Project where there is this need. We have studied FTP and Telnet and describe here an implementation of Fast-TP (Fast Transmission Protocol). After we compare performance tests between Fast-TP and FTP. Finally we present some security aspects of Fast-TP implementation.

1. Introdução

O FTP é um protocolo utilizado em larga escala por todo mundo e se propõe a fazer transferência de arquivos entre cliente e servidor, bem como executar comandos, a partir do cliente, sobre o sistema de arquivos do servidor. Uma comunicação FTP normalmente é composta por uma fase de conexão, onde o cliente é autorizado, e seguida de uma fase de operação na qual arquivos são transferidos e comandos são passados do cliente para o servidor enquanto durar a conexão. Neste tipo de ambiente é comum um ser humano como usuário final na máquina cliente e o tempo gasto com a fase de conexão é irrelevante, não representando um problema para a operacionalização do processo.

Existem casos, no entanto, em que o usuário final no cliente é um aplicativo e não um ser humano, e sua necessidade operacional se resume à transmissão de um ou dois arquivos pequenos de/para o servidor. Neste tipo de ambiente o tempo gasto com o processo de conexão passa a ser extremamente relevante.
Este artigo, partindo de um problema real de um projeto de cartão convênio, propõe um protocolo que melhor atenda ao problema acima explicitado.

Este trabalho está organizado da seguinte forma:
A seção 2 apresenta a um projeto de de cartão convênio e uma breve descrição do Telnet. Na seção 3 detalhamos o protocolo FTP e na seção 4 propomos o Fast-TP. Na seção 5 descrevemos os resultados obtidos com testes do protótipo implementado.

2. Exemplo de um Projeto de Cartão Convênio

Para ilustrar a citação do parágrafo anterior será descrito nesta sessão um projeto de um Sistema de Cartão Convênio em que um aplicativo no cliente que faz uso do protocolo FTP para se comunicar com o servidor e entendemos que esta comunicação poderia ser otimizada se o protocolo Fast-TP fosse utilizado ao invés. Baseado neste exemplo pode-se estender a mesma necessidade para uma diversidade de aplicações imagináveis.
Um projeto de cartão convênio vem da necessidade de mercado de que redes de lojas possam fechar convênios com empresas para que os funcionários das empresas possam comprar na rede. Normalmente as empresas querem ter limite de crédito para seus funcionários e querem receber mensalmente um arquivo de importação sobre as compras efetuadas por seus funcionários segundo algum layout determinado pela empresa. A empresa paga a rede segundo o critério negociado, e desconta em folha de pagamento de seus funcionários segundo sua política interna.
Em um projeto de cartão convênio, um módulo cliente do software da Administradora do cartão recebe requisições de transações por parte de alguma automação comercial (software de farmácia, supermercado, etc), se comunica com o módulo servidor da Administradora do cartão, e retorna para a automação comercial as informações necessárias à efetivação da transação (fig.1).
Está fora do escopo deste trabalho o detalhamento de que tipos de informações transitam entre os dois aplicativos.

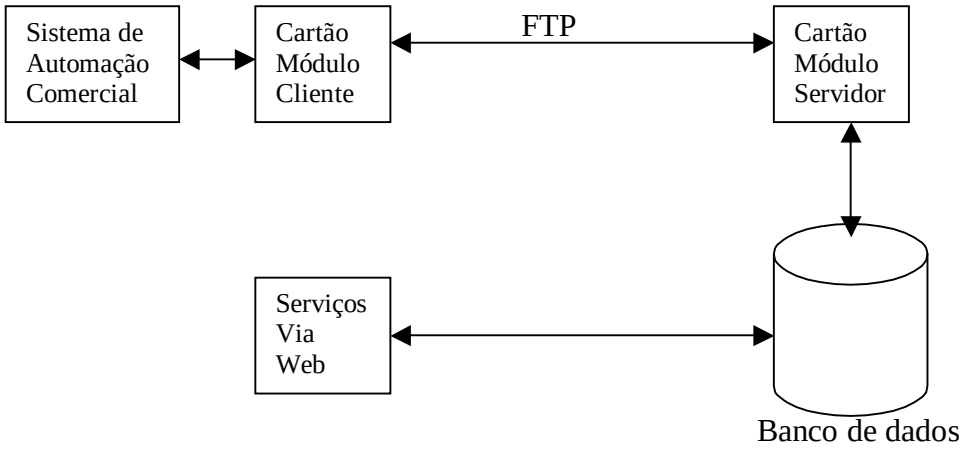

Fig. 1. Exemplo de projeto de cartão convênio.

As informações trafegadas entre os módulos cliente e servidor do cartão são pequenos arquivos texto criptografados com *Blowfish* e autenticados com RC5 compostos de tuplas <parâmetro> = <valor> (fig.2).

Exemplo de conteúdo de arquivo:

****

oper = 10
pinpad = N
pdw = 123
cliente = 00000000000000147864674876 5423
nrterm = 0377734101675500
loja = 03777341016755
valor = 00003646
totvenda = 00003646
produto = 7897930761902; BENZAC WASH 60 gr galderma; ME; 017; 072; N;
0000002048;0000000100;

****

Fig.2. Exemplo de informações transitadas entre os módulos cliente e servidor do Cartão durante uma transação de venda

Como pode-se verificar, na realidade estes arquivos não chegam a conter 1 Kbyte de informação, porém como a unidade mínima de tamanho de arquivo no Microsoft Windows é de 1 Kb, este será o seu tamanho na maioria dos casos.

Na aplicação Cartão Convênio o protocolo de comunicação entre o cliente e o servidor da administradora do cartão em uma operação normal de venda procede como o exemplo abaixo:

- Módulo Cliente envia arquivo .001 com informações pertinentes a autenticação do usuário do cartão e aos produtos e valores da compra desejada.
- Módulo Servidor valida informações recebidas com o banco de dados da administradora (p.ex: Nr de cartão OK, Cliente tem limite de crédito para aquele valor, etc) e retorna arquivo .002 para o cliente com a autorização ou negação da transação pedida.

O tráfego de informações no protocolo acima descrito faz uso do *Safe*-FTP para comunicação cliente-servidor. O *Safe*-FTP é um FTP em que os *Headers* do protocolo (como informação de *User* e Senha do FTP) trafegam cifrados com a chave pública do servidor ou do cliente. Os dados do pacote também podem ser opcionalmente cifrados [5].

Em testes reais efetuados foi observado que o tempo de uma transação Cartão Convênio varia entre 50 a 70 segundos, sendo que cerca de 25 a 30 segundos são gastos com o processo de conexão FTP. Em vista disso acredita-se que o tempo de uma transação Cartão Convênio pode ser otimizado para a faixa de 30 a 50 segundos, o que faz grande diferença operacional nos *check-outs* das lojas que operam este sistema.

3. Principais características do File Transfer Protocol

Os principais objetivos do FTP são: promover o compartilhamento de arquivos, transferir arquivos entre *hosts* e permitir a execução de comandos em computadores remotos [6].

Principais processos envolvidos no FTP:

user-FTP process - Conjunto de funções que inclui um interpretador de protocolo, um processo de transferência de dados e uma interface com o usuário, que juntos executam a função de transferência de arquivos em cooperação com um *server-FTP process.*

Server-FTP-process - Consiste de um interpretador de protocolo e um processo de transferência de dados. Executa a função de transferência de arquivos em cooperação com um *user-FTP process.*

User-PI (User Protocol Interpreter) – Inicia uma conexão de controle a partir de uma porta P, inicia comandos FTP e governa o *User-DTP* se o processo é parte de uma transferência de arquivo.
User-DTP (Data Transfer Process) – fica "escutando" na porta de dados por uma conexão do processo *Server-FTP*.
Server-PI (Server Protocol Interpreter) – "Escuta" na porta P' por uma conexão de um *User-PI*. Recebe comandos FTP de um *User-PI*, envia respostas e governa o *Server-DTP*.
Server-DTP (Data Transfer Process) – Estabelece uma conexão de dados com o *User-DTP*. Define parâmetros para a transferência e armazenamento e transmite dados e comandos a partir de seu PI.

Segundo o modelo do FTP, uma conexão de controle é iniciada pelo *User PI*. A conexão de controle segue o protocolo Telnet. Durante a inicialização, comandos FTP padrão são trocados, através da conexão de controle, entre o *User-PI e o Server-PI*.
Comandos especificam parâmetros para a conexão de dados (porta, modo de transferência, representação de caracteres) e operações a serem executadas no sistema de arquivos remoto. O protocolo exige que uma conexão de controle esteja aberta enquanto existir transferência de dados em andamento. Normalmente é responsabilidade do usuário requerer o fechamento da conexão de controle.
O FTP usa o protocolo Telnet na conexão de controle. Duas maneiras de implementar: na primeira o *User-PI ou o Server-PI* implementam as regras do Telnet diretamente em suas *procedures*; ou na Segunda, *o User-PI ou o Server-PI* fazem uso de módulos Telnet existentes no sistema.

3.1. Estabelecendo Conexão de Dados

A mecânica de transferência de dados consiste em setar a conexão para as portas apropriadas e selecionar os parâmetros para a transferência. Ambos, *user e server-DTP* possuem portas de dados *default*. A porta de dados *default* no *user-process* é a mesma da conexão de controle. No processo servidor a porta de dados *default* será a porta adjacente à porta de controle de conexão (ex. L-1).

3.2. Modos de Transmissão

Existem 3 modos de transmissão: *Stream* Mode onde o arquivo é transmitido como uma sequência de bytes, *Block Mode* onde o arquivo é transmitido em blocos com cabeçalho, e *Compressed Mode* com compressão de dados aumentando a largura de banda.

3.3. Comandos

O canal de comunicação entre o *user-PI* e o *server-PI* é estabelecido por uma conexão TCP do cliente para a porta padrão do servidor. O *user-PI* é responsável por enviar comandos FTP e interpretar as respostas recebidas. O *server-PI* interpreta comandos, envia respostas, e direciona seu DTP para preparar a conexão e a transferência de dados. O FTP deixa o controle de erros de bits perdidos a cargo do protocolo de camada inferior TCP.
O File Transfer Protocol faz uso de várias especificações do Telnet Protocol para as comunicações através da conexão de controle. Os comandos são transferidos como "*Telnet Strings*" segundo a linguagem do Telnet e seguidos de caracteres de fim de linha. Por linguagem Telnet deve-se entender NVT-ASCII e por fim de linha dos comando, <CRLF>.

4. Proposta do Fast Transfer Protocol

O Fast-TP também é baseado em uma estrutura cliente servidor. A idéia inicial do protocolo é procurar abstrair a etapa de conexão Telnet a fim de economizar uma significativa quantidade de tempo.

O FastTP faz uso de sockets e funciona sobre o protocolo TCP (Transmission Control Protocol). Um *socket* consiste, basicamente, de um endereço IP associado a uma porta [2]. Escolhemos a porta 550 para conexão do cliente com o servidor. O servidor funciona escutando esta porta e em seguida estabelece uma conexão com o *socket* do cliente em outra porta enquanto durar a comunicação ou com saída por time-out. O servidor opera em *multi-threads*, sendo capaz de atender a diversas solicitações clientes simultâneas. O Fast-TP funcionará sobre quaisquer meios físicos em que se possa fazer uma transmissão TCP/IP (ex: T1, fio telefônico, Radio freqüência) [10]

Nosso principal problema consiste em garantir autenticação do usuário sem a conexão Telnet.

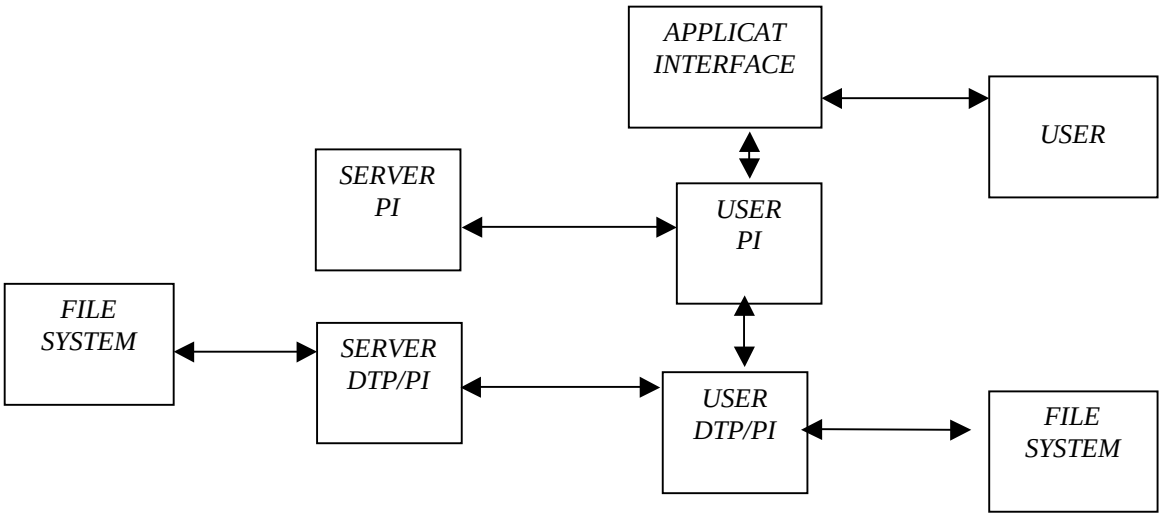

Fig.4. Arquitetura do Fast-TP

O Fast-TP continua com os módulos DTP e PI para fins de transferência de dados e interpretação de comandos suportados, porém substitui a *User Interface* por uma *Application* Interface, uma vez que ele é destinado primariamente a se comportar como componente (objeto) a ser utilizado por aplicações que necessitem de transferências rápidas de arquivos (fig.4). Observe que os outros módulos são essencialmente os mesmos que no FTP tradicional. Continuam sendo necessários os módulos interpretadores de protocolo (PI) em ambas as pontas, bem como os de transferência de dados (DT). Internamente os módulos funcionam diferentemente do FTP tradicional, porém graficamente a representação é similar.

Quanto à representação dos dados, nosso projeto experimental é baseado somente em ambiente Windows, não sendo necessário fazer uso do NVT-ASCII ou qualquer outra tabela de conversão.

A estrutura dos dados pode ser:

File-structure – quando não existe estrutura interna e o arquivo é considerado como sendo uma sequência contínua de bytes;

Como modo de transmissão foi implementado o *Stream Mode* [9].

A etapa de conexão Telnet na realidade é demorada pois é feita uma autenticação e *login* com o sistema operacional. Para a proposta de funcionamento do FTP este *login* é fundamental tendo em vista que o usuário será capaz, além de transferir arquivos, de executar comandos

disponibilizados pelo servidor de FTP e repassados para a Shell do SO (sistema operacional). O Fast-TP, a exemplo do FTP, disponibiliza um conjunto limitado de comandos para com o SO. Uma vez que não existe um login com o SO diferente para cada usuário, optamos por rodar o servidor Fast-TP efetuando automaticamente o *login* dos clientes em uma conta *anonymous* e a disponibilização de comandos da Shell somente pode ser executada sobre arquivos criados pelo próprio usuário (*owner*). Como conseguir isto? Através de um ambiente baseado em infra-estrutura de chave pública [3].
Os pacotes de um arquivo ou de comandos são sempre cifrados com a chave privada do cliente e contem a chave publica do cliente em texto plano para que o servidor possa decifrar o resto do pacote e para fins de autenticação. Um *hash* do arquivo é acrescentado na origem e retirado no destino em qualquer comunicação cliente x servidor, e comparado para validação da integridade do arquivo.

O servidor mantém uma tabela com os caminhos completos para os arquivos e a chave pública do usuário identificando o *owner*.

Esta estrutura simples resolve nosso problema de performance ao evitar o *login* com o SO via Telnet. Resolve também o problema de execução de comandos sobre os arquivos transferidos fazendo com que o software servidor só execute comandos do próprio *owner* do arquivo. Embora deva-se levar em consideração que os algoritmos de criptografia assimétrica normalmente são lentos [4] e o protocolo SSL por exemplo, que é baseado nestes mesmos algoritmos, possui um custo computacional de 5-7 [12], vale lembrar que a proposta deste protocolo é para a transferência de arquivos pequenos, normalmente feita por aplicações.

4.1. Comandos de Serviço

RETRIEVE (RETR) , STORE (STOR) , APPEND (with create) (APPE) , RENAME , ABORT (ABOR) , DELETE (DELE) , REMOVE DIRECTORY (RMD) , MAKE DIRECTORY (MKD) , LIST (LIST).
Dos comandos acima listados foram implementados o STORE e o RETRIEVE para fins de validação deste artigo e os outros estão em fase de implementação.

5. Implementação e Resultados Obtidos

Para validar o Fast-TP foram utilizados uma intranet TCP/IP sobre uma rede local ethernet [1]. Um Servidor Pentium 1000 com 512 Mb RAM e Windows 2000; e três máquinas Clientes Pentium 400 com 128 Mb RAM e Windows 98; O acesso cliente servidor foi feito pela rede local e via internet por uma linha dedicada ADSL da Brasil Telecom.
Os softwares cliente e servidor foram desenvolvidos em Delphi 5 para plataforma windows.

Exemplo de uso do Fast-TP:
Uma aplicação chama o módulo cliente Fast-TP com o comando "Fast-TP Upload testecli.txt testesrv.txt". O módulo cliente entende que deve enviar o arquivo testecli.txt para o módulo servidor e passa para o mesmo o nome testesrv.txt com o qual deverá ser criado no sistema de arquivos do servidor

Os resultados podemos analisar na figura 5.

Tempo médio (em seg) após 10 medições para cada ambiente:

Ambiente	Safe FTP	Fast-TP
Intranet	10 s	< 01 s
Internet	20 s	05 s

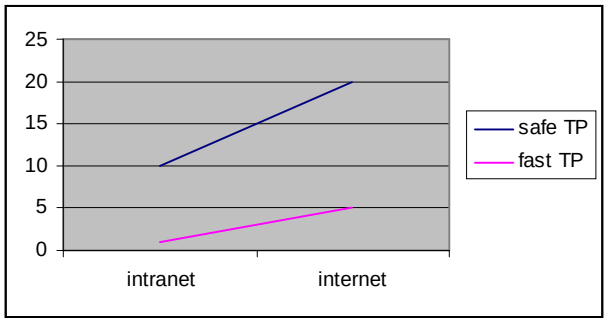

Fig.5. Resultados das medições

Após 10 medições de testes na Intranet, o tempo médio para conexão e envio de um arquivo de 2k utilizando o Safe-FTP foi de 10 (dez) segundos. Já o tempo médio para conexão e envio de um arquivo de 2k utilizando o Fast-TP foi de menos de 01 (um) segundo

Após 10 medições de testes na Internet, Tempo médio para conexão e envio de um arquivo de 2k utilizando o Safe-FTP foi de 20 (vinte) segundos. Já o tempo médio para conexão e envio de um arquivo de 2k utilizando o Fast-TP foi de 05 (cinco) segundos

6. Trabalhos relacionados

RFC 959 [6] já descrito na seção 3.
RFC 854 [7] conforme descrito na seção 2.

The Safe-TP project [5] é um projeto em que existe o software cliente e o servidor Safe-TP. Eles "escutam" a porta de saída/chegada de comunicação FTP e, se detectarem que a outra ponta é um Safe-TP, cifram os dados antes de serem transferidos. Se a outra ponta é um cliente ou servidor FTP comum, o processamento continua sem interferência. O principal problema do safe-TP para nossa necessidade é o problema de ter que fazer uma conexão telnet antes de efetuar operações sobre os arquivos.

O FTP lite provê funcionalidades básicas para clientes Unix que acessem um servidor FTP de cada vez (*single threaded*) [8]. Apesar de um cliente mais simples ele recai no mesmo problema de ter que fazer uma conexão telnet antes de efetuar operações sobre os arquivos, além de não prover confidencialidade aos dados.

*O PES (*privacy enhanced sockets*)* é uma alternativa de segurança para transmissões TCP/IP cliente servidor [11]. A diferença para nossa proposta é que o PES utiliza Diffie-Hellman [13] para fazer uma troca da chave secreta entre cliente e servidor e utilizar criptografia simétrica. Tendo em vista que nossa proposta é para arquivos pequenos, o processo de troca de chaves se torna mais dispendioso do que o ganho de performance da criptografia simétrica em relação a cifrar os dados com criptografia assimétrica usada em nosso processo.

7. Conclusões e Trabalhos Futuros

Nossos testes permitem concluir que, nas condições para as quais o Fast-TP foi proposto ele possui performance superior ao FTP tradicional. Isto foi conseguido eliminando-se a necessidade da etapa de conexão Telnet que faz parte do processo de FTP tradicional.
Por isso o Fast-TP atende melhor ao caso real (descrito na seção 2) de um Projeto de Cartão Convênio. Outras aplicações podem ter necessidades similares.
Em relação a aspectos de segurança: confidencialidade e autenticação são conseguidos com cada usuário tendo um par de chaves pública e privada. Integridade é garantida por uma função *Hash*. Nosso projeto não prevê geração e troca de chave secreta para criptografia simétrica tendo em vista que o Fast-TP se propõe inicialmente a transferências de arquivos pequenos (da ordem de 1 a 2 k), onde mesmo um algoritmo de criptografia assimétrica processará rápido.

Como propostas futuras podemos estabelecer o Fast-TP multiplataforma, baseado na mesma tabela NVT-ASCII de conversão de caracteres usada pelo Telnet.

8. Bibliografia

1. Tanenbaum, Andrew - Redes de Computadores, Editora Campus 1997.
2. Comer, Douglas - Interligação em redes com TCP/IP, Editora Campus 2000.
3. Stallings, William - Cryptography and network security, Prentice Hall 2000.
4. Schneier, Bruce – Applied Cryptography, Wiley 2000.
5. Dan Bonachea, Scott McPeak, *SafeTP: Transparently Securing FTP Network Services*", http://www.cs.berkeley.edu/~smcpeak/SafeTP/, UCB Tech Report CSD-01-1152, February 2001.
6. J. Postel, J Reynolds, RFC959 File Transfer Protocol, http://www.faqs.org/rfcs/rfc959.html, FreeSoft.org, out 1985.
7. J. Postel, J Reynolds, RFC854 Telnet Protocol, http://www.freesoft.org/CIE/RFC/854/, FreeSoft.org, mai 1983.
8. Douglas Thain, Ftp Lite Project, http://www.cs.wisc.edu/condor/ftp_lite/, University of Wisconsin – Computer Science Department, jan 2002.
9. Khare, Rohit, I Want My FTP: Bits on Demand, IEEE Internet Computing, vol.2, nr.4, p.88-91, aug-1998.
10. L.Rodney Long, High Speed Satellite Access to Biomedical text/image database, Advanced Digital Libraries, 1996
11. Zuquete Andre, Guedes Paulo, Transparent Authentication and Confidenciality for stream sockets, Vol. 16, No. 3; JUN 1996, p. 34-41
12. Krishna Kant and Ravishankar Iyer, Arquitectural Impact of Secure Socket Layer on Internet Servers, 2000 IEEE International Conference on Computer Design, 2000
13. W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory,* Vol. IT-22, No. 6, Nov. 1976, p 644-654