Centro Universitário de Brasília
Instituto CEUB de Pesquisa e Desenvolvimento – ICPD

Marcos Victor Boni de Vasconcelos Santos

Análise de Desempenho do Tráfego da informação de uma Simulação VPN Com o Conjunto de Protocolos IPSec

BRASÍLIA
2008

Marcos Victor Boni de Vasconcelos Santos

Análise de Desempenho do Trafego da informação de uma Simulação VPN Com o Conjunto de Protocolos IPSec

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/IPCD) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Graduação, na área de Engenharia da Computação.

Orientador: Prof. Marco Antônio Araújo

BRASÍLIA
2008

Marcos Victor Boni de Vasconcelos Santos

Análise de Desempenho do Tráfego da informação de uma Simulação VPN Com o Conjunto de Protocolos IPSec

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/IPCD) como pré-requisito para a obtenção de Certificado de Conclusão de Graduação, na área de Engenharia da Computação.

Orientador: Prof. Marco Antônio Araújo

Brasília, ___ de_____ de 2008

Banca Examinadora

_____

Prof. Dr....

_____

Prof. Dr....

DEDICATÓRIA

Dedico essa monografia principalmente aos meus pais que me deram o suporte que eu tanto precisava não somente agora na etapa da monografia, mas sim durante todo o meu curso de engenharia.

AGRADECIMENTOS

Agradeço a todos que me ajudaram no desenvolvimento do meu projeto, foram várias etapas a serem vencidas no desenvolvimento da ferramenta de gerência de redes.

Eu devo...
Me deter com as novas regras
Pois nem foram ainda formuladas
E meus pensamentos cantam a plenos pulmões
Na certeza de que eu e meus semelhantes
Seremos aqueles, formuladores das regras...
Se as pessoas de amanhã
Precisarem mesmo das regras de hoje
Então venham, procuradores!
O mundo é um julgamento
Sim (...)

Bob Dylan

RESUMO

A princípio, este trabalho apresenta o desenvolvimento de um túnel VPN entre duas máquinas para em seguida fazer uma análise de desempenho da informação de tudo aquilo que está passando quando o túnel está ligado ou desligado.

Os mecanismos de segurança utilizados nesta monografia estão atribuídos em um conjunto de padrões criptográficos na construção do IPSec – *Internet Protocol Security*, para podermos prover segurança na tramitação da informação junto ao protocolo IPv4 - *Internet Protocol Version* 4, no qual, os cabeçalhos serão apresentados e comparados. Esta monografia visa tratar problemas relacionados à segurança da informação e identificar o desempenho do tráfego relacionado à tramitação da informação antes e após a aplicabilidade do IPSec.

Para a analise de desempenho será coletada as informações trafegadas em cenários diferenciados pelo tamanho do arquivo e armazenadas em um banco de dados para posteriormente observar às curvas do gráfico e atribuir a relação comparativa no tempo gasto na tramitação da informação em claro e da informação criptografada.

LISTA DE FIGURAS

LISTA DE QUADROS E TABELAS

LISTA DE ACRÔNIMOS

ASN.1 – Abstract Syntax Notation One
BER - Basic Encoding Rules
CBC - Cipher-Block-Chaining
CCITT - União Internacional de Telecomunicações
DDL - Data Definition Language
DES - Data Encryption Standard)
DMTF - Desktop Management Task Force
DOS - Disk Operating System
HEMS - High-level Entity Management System
HTTP - HyperText Transfer Protocol
IAB - Internet Archtecture Board
IEC - Comissão eletrotécnico internacional
IP - Internet Protocol - Protocolo de Interconexão
IPsec- Internet Protocol Security
ISO - Organização Internacional para Padronização
NAT - Network Address Translation
RDBMS - Relational Database Management System
SQL - Structured Query Language ou Linguagem de Consulta Estruturada
TCP - Transmission Control Protocol ou Protocolo de Controle de Transmissão
TI - Tecnologia da Informação
UDP - User Datagram Protocol
VPN - Virtual Private Network

SUMÁRIO

1 INTRODUÇÃO

A informação é um recurso de maior relevância para uma pessoa ou organização. Através da utilização da informação pode-se realizar qualquer atividade necessária para existência humana ou empresarial. Hoje, no mundo digital a informação está presente com maior quantidade, qualidade, e rapidez de acesso. Para trafegar a informação no mundo computacional com segurança, isso é minimizar as possibilidades de captura, manipulação ou destruição dos dados trafegados ou armazenados, exigi-se um conjunto de procedimentos a serem executados.

As existências de novas tecnologias favorecem consigo diferentes tipos de vulnerabilidades e possivelmente novos tipos de ataques serão criados. O aumento da conectividade de acesso na internet e o conjunto de novos protocolos de comunicação podem trazer também diferentes tipos de ameaças. Com a expansão da internet, o aumento dos crimes digitais está cada vez mais organizado, pois além do limite geográfico transcender fronteiras, a legislação para atuar nas ações perversas não atingem uma maturidade normativa para inibir os criminosos em diversos países.

Para suprir alguns dos diversos tipos de vulnerabilidades, ameaças ou ataques são implementados mecanismos de segurança voltados à criptografia dos dados. É importante que se tenha uma análise do desempenho dos métodos utilizados para envolver a segurança da informação.

Os mecanismos de segurança utilizados nesta monografia estão atribuídos a um conjunto de padrões criptográficos na construção do IPSec – *Internet Protocol Security*, para que possa prover segurança na tramitação da informação junto ao protocolo IPv4 - *Internet Protocol Version* 4, no qual, os cabeçalhos serão apresentados e comparados. Esta monografia visa tratar problemas relacionados à segurança da informação e identificar o desempenho do tráfego relacionado à tramitação da informação após a aplicabilidade do IPSec.

Inclui-se neste trabalho a instalação do IPSec em 2 máquinas (A e B), onde é simulado o tráfego criptografado do ponto A para B. Posteriormente é utilizada uma

ferramenta de captura dos dados trafegados, máquina C. O trabalho apresenta uma ferramenta de análise para avaliar o desempenho da comunicação com a utilização do IPSec. Os dados capturados serão armazenados em um banco de dados. Posteriormente será apresentado em página WEB os resultados do desempenho da comunicação quando utilizado o IPSec.

2 SEGURANÇA NO TCP/IP

Este capítulo apresenta os protocolos fundamentais para a comunicação no mundo da internet, o IPv4 - *Intenet Protocol* versão 4 e o TCP - Transmission Control Protocol. A importância da demonstração desses protocolos, consiste posteriormente no capítulo 3 com a utilização do IPSec – Internet Protocol Security encapsulado junto ao cabeçalho IP. Apresentam-se também algumas vulnerabilidades que a informação pode sofrer no mundo da internet e os mecanismos de defesas que podem ser exercidos para prover a tramitação segura da informação.

2.1 Existência dos protocolos TCP/IP

Para existência da internet há a necessidade de utilizar dois protocolos padrões de comunicação que se complementam TCP/IP. Esses protocolos foram desenvolvidos pela DARPA (Defense Advanced Research Project Agency) no DoD (Departamento de Defensa dos Estados Unidos).

Este conjunto de protocolos TCP/IP foi desenvolvido para permitir aos computadores compartilharem recursos numa rede. Toda a família de protocolos inclui um conjunto de padrões que especificam os detalhes de como comunicar computadores, assim como também convenções para interconectar redes e rotear o tráfego (COMER 1998).

O protocolo IP pertence à camada de rede, pode ser considerado um protocolo de extrema importância já que outros protocolos, tais como TCP, UDP e ICMP são transmitidos pelo datagrama IP. Este protocolo contém informações sobre endereçamento e algumas informações de controle. Ele foi definido na RFC 791.

A principal característica do protocolo IP é que a transmissão é efetuada sem a necessidade de uma conexão direta entre *host* fonte e *host* destino, sendo baseada no envio de pacotes de informações que podem passar por muitas redes intermediárias até chegarem ao destino. Porém como citado anteriormente, o

protocolo IP não é confiável, ou seja, ele não garante que o datagrama enviado chegou ao seu destino. Qualquer confiabilidade requerida deve ser feita pelas camadas superiores, como por exemplo, o TCP. O protocolo IP básico oferece três serviços:

a) Entrega sem conexão física, algumas vezes chamada serviço datagrama.

b) Um mecanismo para fragmentação e remontagem do pacote.

c) Endereçamento e roteamento de pacotes (COMER 1998).

A principal característica do protocolo TCP que trabalha na camada de transporte é ser orientado a conexão, garantido assim a entrega do pacote.

2.1.1 Formato do Datagrama IP

Cada datagrama IP consiste de um cabeçalho e uma área de dados. O cabeçalho ocupa uma área fixa de 20 *bytes* e uma área de tamanho variável (correspondente ao campo *options*). Detalhe do cabeçalho IP na Figura 2.1.

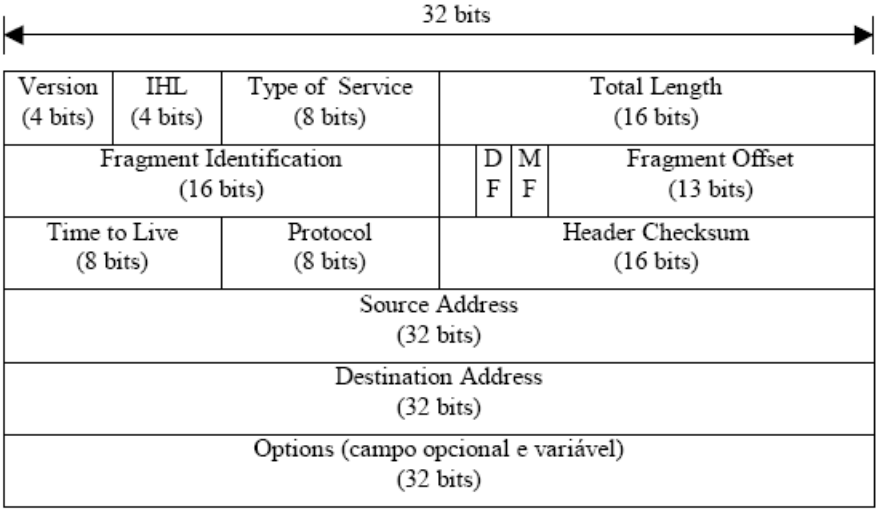

Figura 2.1 - Cabeçalho IP

Fonte: Comer 1998

Abaixo se observam as definições de cada campo do cabeçalho IP, de acordo com Rhee (2003):

a) ***Version***: 4 bits os quais contém a versão do protocolo IP que o datagrama pertence. Ele é utilizado pra verificar se o transmissor, o receptor e quaisquer roteadores existentes entre eles concordam quanto ao formato do datagrama. Todo *software* IP precisa verificar o campo de versão antes de processar um datagrama, para assegurar-se de que ele se adapta ao formato que o software espera. A versão atual é a 4 a qual já está sendo utilizada desde 1981, porém a versão 6 já existe.

b) ***Header Length* (IHL)**: O comprimento do Cabeçalho IP medido em unidades de 32 bits. Este campo não inclui o campo de dados, de modo que é realmente uma medida do comprimento do campo de Opções, uma vez que os outros campos dentro do cabeçalho IP têm comprimento fixo.

c) ***Type of Service* (TOS)**: No RFC 791, os 8 bits seguintes são alocados para um campo tipo de Serviço (ToS) agora DiffServ e ECN. A intenção original era para um host especificar uma preferência para como os datagramas poderiam ser manuseados assim que circulariam pela rede. Por exemplo, um host pode definir o campo de valores do seu ToS dos datagramas IPv4 para preferir pequeno desfasamento de tempo (ou "delay"), enquanto que outros podem preferir alta fiabilidade. Na prática, o campo ToS não foi largamente implementado. Contudo, trabalho experimental de pesquisa e desenvolvimento se focou em como fazer uso destes oito bits. Estes bits têm sido redefinidos e mais recentemente através do grupo de trabalho do DiffServ na IETF e pelos pontos de código do Explicit Congestion Notification (ECN) codepoints (ver RFC 3168), conforme a Figura 2.2**:**

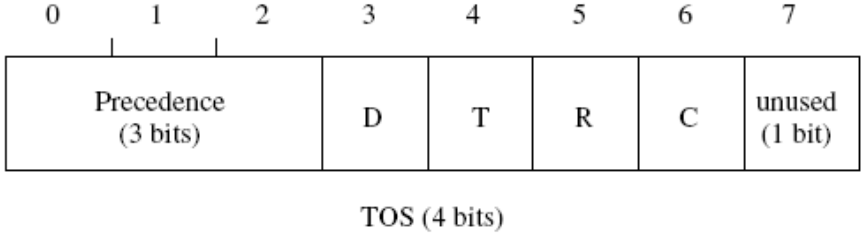

Figura 2.2 - Detalhe do Campo *Type of service*

Fonte: Comer 2000.

Os três bits do campo de precedência especificam a precedência do datagrama com valores variando de 0 a 7, permitindo que os transmissores indiquem a importância de cada datagrama. Embora a maioria dos *softwares* IP ignorem o campo *precedence*, ele pode ser importante, já que fornece um mecanismo que pode permitir que informações de controle tenham precedência sobre dados. Como por exemplo, se todas as estações e roteadores reconhecem a precedência, é possível implementar algoritimos de congestionamento que não sejam influenciados pelo congestionamento que estão tentando controlar.

Os bits D,T, R e C especificam o tipo de transporte que o datagrama deseja. Quando ajustado, o bit D solicita um intervalo baixo, o bit T solicita *throughtput* alto, o bit R solicita alta confiabilidade e o bit C um custo menor. Permitindo várias combinações de velocidade e segurança. Por exemplo: para voz digitalizada, a velocidade é muito mais importante que segurança. Já para transferência de arquivos, uma transmissão com segurança é muito mais importante que a velocidade. Veja que ainda ha um bit o qual não é utilizado e este dever ter o bit 0.

a) ***Total length***: O campo de 16 bits seguinte do IPv4 define todo o tamanho do datagrama, incluindo cabeçalho e dados, em bytes de 8 bits. O datagrama de tamanho mínimo é de 20 bytes e o máximo é 65535 (64 Kbytes). O tamanho máximo do datagrama que qualquer host requer para estar apto para manusear são 576 bytes, mas os hosts mais modernos manuseiam pacotes bem maiores.

b) **ID**: Identifica unicamente cada datagrama. Muito útil nas remontagens de datagramas fragmentados já que todos os fragmentos de um datagrama contêm o mesmo valor de identificação, é utilizado para a estação destino identificar o datagrama que pertence cada fragmento.

c) **DF**: *Don't Fragment* - aviso para os roteadores não fragmentarem o datagrama, pois o destino não é capaz de remontá-los novamente, um aplicativo pode optar por não permitir uma fragmentação quando somente o datagrama inteiro é útil. Se este datagrama não puder passar através da rede física ele é descartado e uma mensagem ICMP (*Internet Control Message Protocol*) é enviada para a estação que a originou.

d) **MF**: *More Fragments* - todos os fragmentos, com exceção do último, devem possuir este bit ligado. Ele é utilizado juntamente com o campo total *length* para garantir que nenhum fragmento esteja faltando. Quando essa *flag* for 0, quer dizer que se trata do último fragmento ou que o pacote não foi fragmentado.

e) ***Fragment offset***: indica a posição à qual pertence o fragmento atual, permitindo que os fragmentos dos datagramas cheguem aos pontos de montagem fora de seqüência e que sejam retidos temporariamente num *buffer* para aguardar fragmentos restantes. Todos fragmentos com exceção do último devem ser múltiplos de 8 bytes (unidade básica do fragmento). Como são utilizados 13 bits, há um máximo de 8192 fragmentos por datagrama, dando um tamanho máximo de 65536 bytes, coerentemente com o campo total *length*.

f) ***Time to live***: Um campo de 8 bits, o TTL (*time to live*, ou seja, tempo para viver) ajuda a prevenir que os datagramas persistam (ex. andando aos círculos) numa rede. Historicamente, o campo TTL limita a vida de um datagrama em segundos, mas tornou-se num campo de contagem de hops. Cada switch de pacotes (ou router) que um datagrama atravessa decrementa o campo TTL em um valor. Quando o campo TTL chega a zero, o pacote não é seguido por um switch de pacotes e é descartado.

g) ***Protocol***: Um campo de Protocolo de 8 bits segue-se. Este campo define o protocolo seguinte usado numa porção de dados de um datagrrama IP. A Internet Assigned Numbers Authority mantém uma lista de números de protocolos. Os protocolos comuns e os seus valores decimais incluem o Protocolo ICMP (*Internet control message protocol*, ou seja, Protocolo de controlo de mensagens da Internet) (1), o Protocolo TCP (*Transmission Control Protocol*, ou seja, Protocolo de controlo de transmissão)

h) ***Header checksum***: O campo seguinte é um campo de verificação (*checksum*) para o cabeçalho do datagrama IPv4. Um pacote em trânsito é alterado por cada router (*hop*) que atravesse. Um desses routers pode comprometer o pacote, e o *checksum* é uma simples forma de detectar a

consistência do cabeçalho. Este valor é ajustado ao longo do caminho e verificado a cada novo *hop*. Envolve apenas verificação do cabeçalho (não dos dados).

i) ***Source address***: indica o número da rede e número do *host* origem da mensagem.

j) ***Destination address***: indica o número da rede e número do *host* destino da mensagem.

k) ***Options***: Campos do cabeçalho adicionais (chamados de *options*, opções) podem seguir o campo do endereço de destino, mas estes não são normalmente usados. Os campos de opção podem ser seguidos de um campo de caminho que assegura que os dados do utilizador são alinhados numa fronteira de words de 32 bits. (No IPv6, as opções movem-se fora do cabeçalho standard e são especificados pelo campo Next Protocol, semelhante à função do campo "Protocolo" no IPv4). A seguir, três exemplos de opções que são implementadas e aceitas na maioria dos roteadores:

2.1.2 Endereçamento IP

Os endereços IP têm 32 bits de comprimento, normalmente escritos como quatro octetos (em decimal), por exemplo 192.168.1.66. A primeira parte do endereço identifica uma rede especifica na inter-rede, a segunda parte identifica um *host* dentro dessa rede. Os endereços IP compreendem o conceito de Classe de Endereço. A classe de um endereço IP decide quanto do endereço será interpretado como Identificação de Rede.

Endereçamento Classe A (Figura 2.3)

a) primeiro byte identifica a rede

b) bytes restantes identificam o *host*

c) valores de rede de 0 e 127 são reservados

d) há 126 redes de classe A

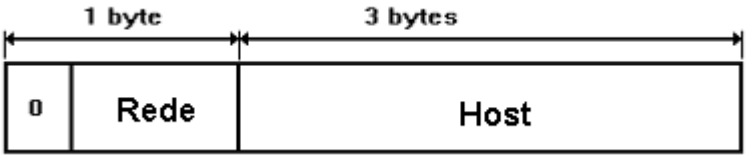

Figura 2.3 - Endereçamento Classe A

Fonte: Torres 2001.

Endereçamento Classe B (Figura 2.4)

a) os primeiros dois bytes identificam a rede

b) os últimos dois bytes identificam o *host*

c) há mais de 16 mil redes classe de B

d) há 65 mil *hosts* em cada rede classe de B

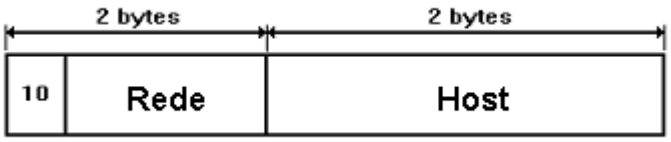

Figura 2.4 - Endereçamento Classe B

Fonte: Torres 2001.

Endereçamento Classe C (Figura 2.5)

a) os primeiros três *bytes* identificam a rede

b) último *byte* identifica o *host*

c) há mais de 2 milhões de redes classe de C

d) há 254 *hosts* em cada rede classe de C

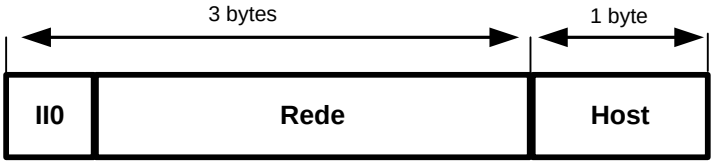

Figura 2.5 - Endereçamento Classe C

Fonte: Torres 2001.

2.1.2.1 Máscara de rede

Como já foi visto, o endereçamento IP consiste de um prefixo o qual determina a rede e um sufixo que significa o endereço do *host* nesta rede. Todo endereço IP tem associado a ele uma máscara de sub-rede, máscara esta que serve para definir quantos bits do endereço IP são relativos a endereços de rede. Os bits da máscara de sub-rede são definidos com um, caso trate o bit correspondente, do endereço IP, como parte do endereço da rede, e zero caso ela trate parte do identificador do *host*, conforme exemplo a seguir:

111111111	11111111	11111111	00000000	/	255.255.255.0

Na máscara acima, os três primeiros octetos identificam a rede, enquanto o quarto octeto identifica um *host* nesta rede. Conseqüentemente esta é uma máscara de classe “C”.

A máscara da rede também pode ser customizada, esta técnica é bastante utilizada para evitar o desperdício de endereço IP. Através da customização você pode fazer a divisão de uma rede padrão (com máscara de 8, 16 e 24 bits) em várias sub-redes. Isto se faz colocando os bits que antes eram alocados para o endereço de *host*, portanto, estavam com valor 0, em 1, ou seja os bits que antes identificavam uma estação passam a ser parte integrante da identificação da rede. No exemplo acima vimos uma máscara de 24 bits, se quisermos dividir esta única rede em sub-redes devemos alterar o número de bits iguais a 1 na máscara de sub-rede. Portanto, ao invés de 24 bits, deverá utilizar 25, 26, 27 ou um número a ser definido.

2.2 Vulnerabilidades e Tipos de ataques

A informação é a principal fonte de riqueza que podemos obter, sendo ela empresarial ou pessoal. Os dados trafegados na internet ou armazenados nos servidores ou estações pessoais devem ser tratados de modo onde as informações tenham garantia de confiabilidade, integridade e disponibilidade.

As vulnerabilidades atingem diversos âmbitos da informação, tais como:

a) Servidor não atualizado pode abrir brechas de acessos indevidos, captura de informação sigilosa, permissão de penetração de vírus.

b) Senhas fáceis podem permitir que usuários indesejados acessem o sistema e tenha controle, manipulação dos arquivos e do sistema no computador.

c) Inexistência de uma política de segurança em uma empresa ou política mal formulada pode acarretar em falhas de segurança da informação.

d) Tramitação da informação para os meios externos sem devida proteção pode expor os dados a serem capturados, manipulados e corrompidos.

Mesmo com os mecanismos de segurança, diversos tipos de ataques ameaçam as vulnerabilidades.

2.2.1 Tipos de Ataques

Com a abrangência cada vez maior dos sistemas computacionais e sua utilização é necessário compreender as técnicas e ferramentas de ataque para que se possa detectar e lidar com os ataques. O termo comum utilizado para identificar quem executa o ataque em um sistema computacional é o *hacker.* Bem como seus diferentes objetivos, entre eles, destruição, manipulação ou captura da informação o sucesso depende do grau de segurança implementado. A seguir identificaremos alguns tipos de ataques que possam comprometer uma pessoa ou organização.

2.2.1.1 Packet Sniffing

Também conhecida como *passive eavesdropping*, essa técnica consiste na captura de informações diretamente pelo fluxo de pacotes que trafegam no mesmo seguimento da rede em que o software *sniffer* funciona. Diversos tipos de filtros podem ser utilizados para captura de pacotes específicos referentes a determinados endereços IP, serviços ou conteúdo. (NAKAMURA, GEUS 2007).

O *dsniff* é um *sniffer* de se senha escrita por Dug Song. Ela entende vários serviços diferentes que transmitem informações de senha em texto claro, além de outras se você lhe der a chave apropriada (CHESWICK, BELLOVIN e RUBIN, 2003).

O *packet sniffing*, que consiste na captura de pacotes que circulam na rede, podendo conter informações importantes e, portanto, confidenciais para a empresa, tais como segredos de negócio e senhas de sistemas de *software*. Os serviços de FTP – File Transfer Protocol e *Telnet* são vulneráveis a esse tipo de ataque, pois é possível obter facilmente as senhas dos usuários que utilizam esses serviços. Neste trabalho utilizaremos uma técnica de *sniffing* com a ferramenta *Wireshark* para podermos capturar e tentar identificar a informação trafegada.

2.2.1.2 Captura de Usuário e Senha nas seções FTP e Telnet

As sessões FTP e Telnet trafegam com seus usuários e senhas em claro permitindo com que um hacker identifique o acesso e posteriormente façam uso do meios autenticação para se passar pelo usuário.

A utilização de *sniffings* é um método comum para a captura de usuários e senhas das sessões FTP e Telnet.

2.2.1.3 TCP SYN Flood Ataque

Segundo Bezerra (2004), o TCP SYN ataque tira vantagem do comportamento de 3 WAY HANDSHAKE efetuado pelo protocolo TCP no processo de uma conexão. O atacante faz um pedido de conexão para o servidor de vitima com pacotes que carregam o endereço falsificado do IP de origem (método IP spoofing). Como resultado o servidor da vitima perde tempo e recursos de máquina que poderiam estar sendo usados para outros processos. Dependendo da quantidade de pacotes de TCP SYN enviado para o servidor da vítima a memória, CPU e aplicações rodando neste servidor serão comprometidos.

2.2.1.4 TCP Session Hijacking

TCP *session hijacking* também conhecido como ataque *man in the middle,* é possivel quando um *hacker* toma controle de uma sessão TCP entre duas maquinas. Normalmente a autenticação ocorre somente no início de uma sessão TCP conforme figura 2.6, isso permite que um *hacker* ganhe o acesso da maquina.

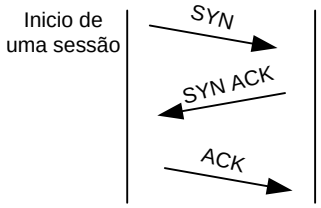

Figura 2.6 - Inicio de uma sessão TCP

Fonte: Autor 2008

Uma técnica comum é utilizar os pacotes IP do roteamento de origem. Isso permite com que o *hacker* se infiltre entre um ponto de comunicação de A para B fazendo com que os pacotes IP passem através de sua maquina.

Utilizando um programa de *sniffing* para capturar a conversa o *hacker* pode também ficar entre A e B e identificar os dados trafegados.

2.2.1.5 IP Spoofing

O IP *Spofing* é um método de ataque que consiste na criação do pacote IP utilizando endereço IP de outra origem. É uma técnica utilizada para mascarar o endereço IP do remetente falsificando-o, proporcionando alterações do endereço IP de origem pelo endereço IP de uma máquina que está dentro da rede.

2.3 Segurança de Dados

Após identificarem-se as possibilidades e técnicas de ataques dentro dos diversos objetivos para capturar, manipular e destruir a informação deve-se considerar os métodos de defesa. Além de resguardar a informação com políticas adequadas de acesso aos dados e atualização de sistema é importante precaver com a tramitação da informação. Para trafegar os dados com devida segurança é utilizado mecanismos de criptografia para assegurar a integridade, confidencialidade e o não repúdio da informação.

A implementação de mecanismos de segurança minimiza as chances de ocorrerem vulnerabilidades para serem exploradas e facilita a administração das redes e recursos de forma segura. É importante frisar que este conjunto representa o mínimo indispensável dentro de um grande universo de boas práticas de segurança, o que equivale dizer “que a sua adoção é um bom começo, mas não, necessariamente, suficiente em todas as situações (Ferreira 2003).

Ainda conforme o autor a segurança da informação é caracterizada pela preservação de:

a) Confidencialidade: garantia de que a informação é acessível somente por pessoas autorizadas;

b) Integridade: salvaguarda da exatidão da informação e dos métodos de processamento;

c) Disponibilidade: garantia de que os usuários autorizados obtenham acesso à informação e aos ativos correspondentes sempre que necessário;

d) Autenticidade: consiste na garantia da veracidade da fonte das informações. Por meio da autenticação é possível confirmar a identidade da pessoa ou entidade que presta a informação;

e) Identificação e Autenticação: distinguir, determinar e validar a identidade do usuário/entidade (se é quem diz ser);

f) Não-repúdio: impedir que seja negada a autoria ou ocorrência de um envio ou recepção de informação.

2.3.1 Criptografia

Entende-se o processo de tornar a informação ilegível por "encriptar", enquanto "desencriptar" é o processo inverso, ou seja, retornar o código para algo compreensível. A ciência de inventar códigos denomina-se "criptografia", enquanto a ciência de decifrá-los chama-se "criptoanálise" (Volpi 2001).

Moreno (2005) afirma que o algoritmo de criptografia é uma seqüência de procedimentos que envolvem uma matemática capaz de cifrar e decifrar dados sigilosos. O algoritmo pode ser executado por um computador, por um hardware dedicado e por um humano. Em todas as situações, o que diferencia um de outro é a velocidade de execução e a probabilidade de erros. Existem vários algoritmos de criptografia.

Além do algoritmo, é utilizada uma chave. Na criptografia computadorizada, a chave é um número, ou um conjunto de números, que protege a informação cifrada. Para decifrar o texto cifrado, deve o algoritmo ser alimentado com a chave correta, que é única. Na figura 2.7, é ilustrado o esquema geral de cifragem utilizando chave.

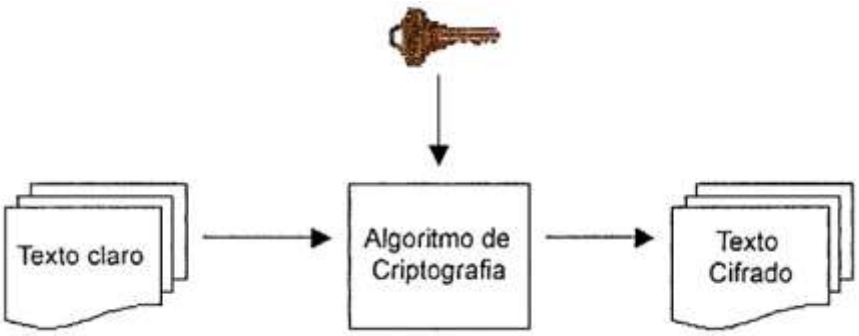

Figura 2.7 - Esquema geral de cifragem com chave
Fonte : Moreno 2005

Com o passar dos anos, os criptoanalistas evoluíram na ciência de decifrar as chaves secretas. Com isto, criou-se a necessidade de uma evolução constante nos métodos de cifragem (criptografia). A partir da evolução dos meios de criptografia, podem ser encontrados dois diferentes processos de cifragem: a criptografia simétrica (convencional) e a criptografia assimétrica (chave pública) (Volpi 2001).

2.3.1.1 Criptografia Simétrica (Convencional)

A criptografia simétrica é utilizada em um criptossistema em que a criptografia e a decriptografia são realizadas usando a mesma chave. Ela também é conhecida como criptografia convencional. É importante ressaltar que a criptografia simétrica transforma o texto claro em texto cifrado, usando uma chave secreta e um algoritmo de criptografia. Usando a mesma chave e um algoritmo de decriptografia, o texto claro é recuperado a partir do texto cifrado (STALLINGS 2008).

Ainda o autor afirma que é impraticável decriptografar uma mensagem com base no texto cifrado mais o conhecimento do algoritmo de criptografia/decriptografia, ou seja, não há necessidade de se manter o algoritmo secreto, conforme pode ser observado na figura 2.8.

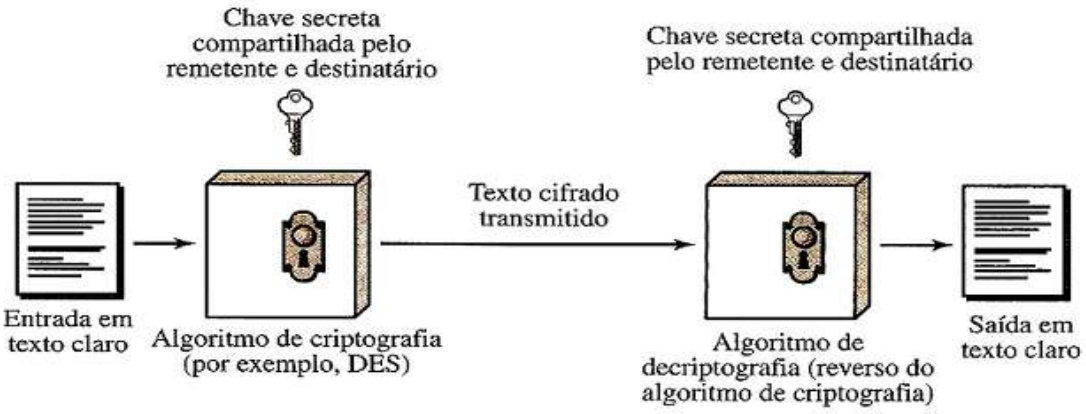

Figura 2.8 – Modelo simplificado da criptografia convencional

Fonte: Stallings 2008

Nakamura e Geus (2007) afirmam que os algoritmos de chave simétrica têm como característica a rapidez na execução, porém eles não permitem a assinatura e a certificação digital. Além disso, existe o problema da necessidade de distribuição das chaves secretas a serem utilizadas pelos usuários, que deve ser feita de maneira segura. O problema está na dificuldade de enviar a chave gerada para o usuário, pois o canal de comunicação ainda não é seguro. Outro problema é o uso de chaves secretas diferentes para cada tipo de comunicação e também para cada mensagem, o que faz com que seu gerenciamento se torne muito complexo. Um exemplo dessa complexidade pode ser visto em um ambiente no qual três usuários se comunicam entre si, onde cada um deles deve armazenar e gerenciar três chaves diferentes. A figura 2.9 mostra que Maria precisa de três chaves secretas diferentes para se comunicar com João, Pedro e Luís.

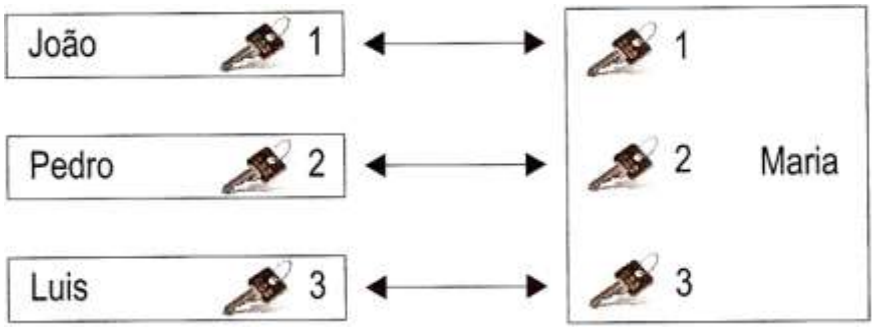

Figura 2.9 – As chaves secretas necessárias na criptografia simétrica

Fonte: Nakamura e Geus 2007

Podemos citar o algoritmo de criptografia DES - *Data Encryption Standard* como criptografia simétrica. O DES foi desenvolvido na década de 70 pelo *National*

Bureau of Standarts com ajuda da *National Security Agency*. O propósito era criar um método padrão para proteção de dados. A International Business Machines – (IBM) criou o primeiro rascunho do algoritmo, chamando-o de LUCIFER. O DES tornou-se oficialmente norma federal americana em novembro de 1976.

Com um tamanho de chave de 56 bits, existem 256 chaves possíveis, o que é aproximadamente 7,2 x 1016 chaves. Assim, um ataque de força bruta parece ser impraticável. Supondo que, na metade do espaço da chave tenha de ser pesquisada, uma única máquina realizando uma criptografia DES por microssegundos levaria mais de mil anos para quebrar a cifra (STALLINGS 2008).

Conforme RAPPAPORT o algoritmo DES trabalha com 64 bits de dados a cada vez. Cada bloco de 64 bits de dados sofre de 1 a 16 iterações (16 é o padrão DES). Para cada iteração um pedaço de 48 bits da chave de 56 bits entra no bloco de ciframento. A deciframento é o processo inverso. O DES pode ser quebrado por ataque de força bruta.

2.3.1.2 Criptografia Assimétrica (Chave Pública)

Os algoritmos assimétricos diferente dos simétricos contam com uma chave para criptografia e uma chave diferente, porem relacionada para decriptografia.

Segundo Rappaport 2003, as chaves assimétricas são um pouco mais complicadas, mas muito mais fáceis de gerenciar. Elas permitem que a informação seja encriptada por uma chave e decriptada por outra. As duas chaves utilizadas nesse cenário são chamadas de chave pública, que é distribuída, e chave privada, que deve ser mantida em segredo. Com chaves assimétricas, os parceiros no negócio trocam suas chaves públicas para se comunicar, mas mantém suas chaves privadas em segredo.

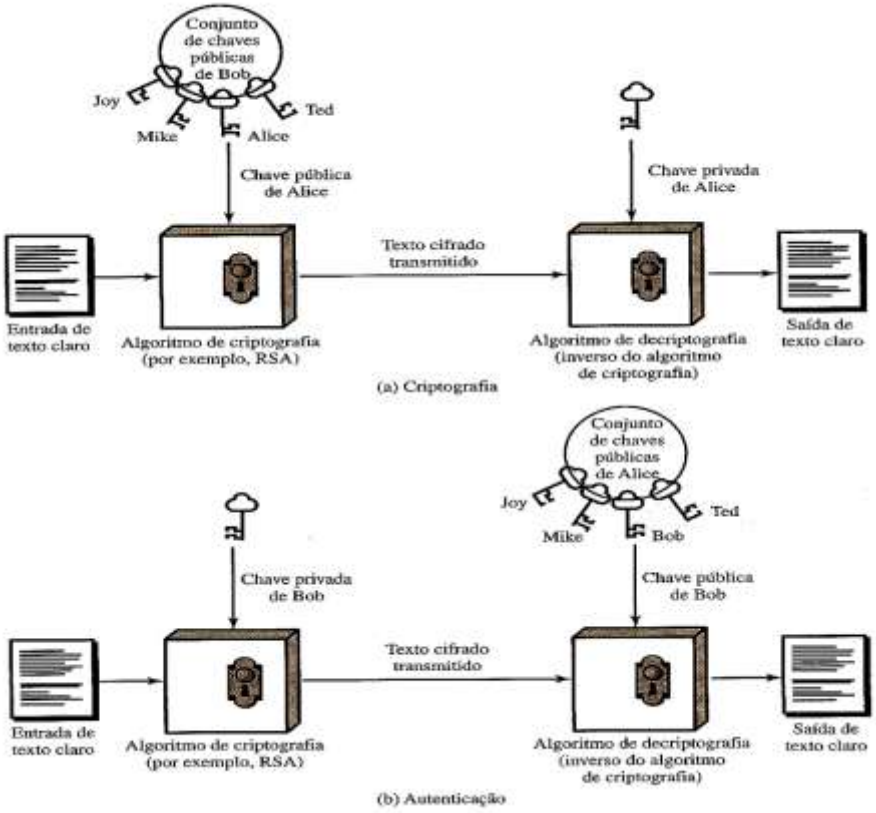

Figura 2.10 – Criptografia de chave pública

Fonte: Stallings 2008

Conforme figura 2.10, a criptografia com chave pública é um esquema assimétrico que usa um par de chaves para ciframento: uma chave pública, que encripta dados, e uma chave privada correspondente (as duas chaves são relacionadas entre si), secreta para deciframento. Diz-se que a sua chave pública não é para todos enquanto mantém sua chave privada em segredo. Qualquer um com uma cópia de sua chave pública pode, então, encriptar informação que só você pode ler. É computacionalmente quase impossível deduzir a chave privada a partir da chave pública.

Conforme quadro comparativo 1, o benefício primário de criptografia com chave pública, é que, permite as pessoas que não têm nenhum arranjo de segurança, possam trocar mensagens com segurança. A necessidade do transmissor e receptor para compartilhar chaves secretas por algum canal seguro é eliminado, todas as comunicações só envolvem chaves públicas, e nenhuma chave

privada é transmitida ou é compartilhada. Um exemplo de sistema de criptografia com chave pública é o RSA[1].

Criptografia convencional	Criptografia de chave pública
Necessário para funcionar: 1. O mesmo algoritimo com a mesma chave é usado para criptografia e decriptografia. 2. O emissor e o receptor precisam compartilhar o algoritmo e a chave **Necessário para a segurança:** 1. A chave precisa permanecer secreta 2. Deverá ser impossível ou pelo menos impraticável decifrar uma mensagem se nenhuma outra informação estiver disponível. 3. O conhecimento do algoritmo mais amostra do texto cifrado precisam ser insuficientes para determinar a chave.	**Necessário para funcionar:** 1. Um algoritmo é usado para criptografar e decriptografar com um par de chaves, uma para criptografia e outra para decriptografia. 2. O emissor e o receptor precisam ter uma das chaves do par casado de chaves (não é a mesma chave). **Necessário para a segurança:** 1. Uma das duas chaves precisa permanecer secreta 2. Deverá ser impossível ou pelo menos impraticável decifrar uma mensagem se nenhuma outra informação estiver disponível. 3. O conhecimento do algoritmo mais uma das chaves mais amostras do texto cifrado precisam ser insuficientes para determinar a outra chave.

Quadro 1 - Criptografia convencional e de chave pública

Fonte: Stallings 2008

Observa-se que a grande vantagem da chave simétrica é sua velocidade em relação à chave assimétrica. Em relação à desvantagem de compartilhar a chave simétrica é a necessidade de uma cópia em cada extremidade. A chave simétrica não pode ser usada para finalidades de autenticação, então, seu algoritmos é raramente usados sozinho.

Neste capítulo podem-se observar os protocolos de comunicação da internet bem como seu funcionamento e suas vulnerabilidades exigindo-se então certos mecanismos de defesa para se evitar comprometimento da informação trafegada ou armazenada, na qual é o principal recuso pessoal ou de uma organização. Nesta monografia será atribuído o conjunto de chaves apresentadas para criação do IPSec.

[1] RSA – Corresponde às iniciais dos sobrenomes dos inventores do código de criptografia, Rivest, Shamir e Adleman.

3 IPSec – INTERNET PROTOCOL SECURITY

Neste capítulo serão abordados os aspectos tecnológicos voltados ao tunelamento do meio de comunicação IPv4 para que a segurança da informação seja aplicada na tramitação dos dados entre um ponto ao outro. Dentre os métodos de tunelamento pode-se citar o conjunto de padrões IPSec sendo utilizado para criação de uma rede privada virtual – VPN. Identifica-se que para criação de uma VPN uma ou mais camadas de protocolos são repetidas de maneira que o canal virtual fique inserido sobre o canal físico. Em um túnel pode-se conectar vários aplicativos, serviços, usuários ou redes pelo canal existente de modo independente sem a necessidade de passar novo cabeamento. Apresenta-se também os mecanismos de defesa IPSec protegendo o protocolo TCP/IP.

3.1 Funcionamento IPSec

O IPSec é um padrão da IETF – *Internet Engineering Task Force,* que foi apresentado em 1998 na RFC 2401, desenvolvido para prover segurança na camada de IP tanto para o protocolo IPv4 que não dispõe de um mecanismo de segurança em sua versão, quanto para o protocolo IPv6 no qual é uma versão robusta em relação a escassez de endereços IPs, mobilidade e segurança. Neste tópico identifica-se a funcionalidade do conjunto de padrões para o funcionamento do IPSec .

Conforme RFC 2401 o conjunto de serviço de segurança que o protocolo IPSec pode prover incluem controle de acesso, integridade na conexão, rejeição de pacotes repetidos, autenticação da origem dos dados e confidencialidade. Em razão desses serviços trabalharem na camada de IP, eles podem ser usados por qualquer protocolo na camada superior em utilidade dos seus serviços TCP, UDP, ICMP, BGP, etc...

O IPSec utiliza de dois protocolos para prover a segurança no tráfego de informações AH e ESP:

a) O AH - *Authentication Head,* provê a autenticação e integridade dos pacotes entre origem e destino, mas não a confidencialidade. O Header de autenticação é utilizado para serviços de autenticação. O AH pode operar isolado em conjunto com o ESP ou embutido quando o modo de tunelamento é usado. A autenticação suprida pela AH difere da ofertada pelo ESP na medida em que este não protege os endereços de IP de origem e destino. Os serviços do header de autenticação matem a privacidade de todo o pacote, inclusive do endereço IP de origem. O AH não protege todas as informações do header externo do IP porque os endereços podem variar durante a propagação do pacote pela rede e não há uma forma de antecipar qual o comportamento dos routers de trânsito. O AH protege tudo o que não pode ser modificado durante o transporte do pacote entre origem e destino (GOLDANI 2004).

b) ESP - *Encapsulating Security Payload,* provê autenticação, confidencialidade dos dados e integridade da mensagem. O campo de autenticação do ESP contém um Integrity Check Value (ICV) que é acrescido à assinatura digital calculada sobre a parte restante do ESP. O ICV tem tamanho variável dependendo do algoritmo de autenticação utilizado. A autenticação é calculada no pacote ESP depois da ciframento estar concluída. O padrão atual do IPSec utiliza a HMAC (assinatura assimétrica) que é validada por algoritmos SHA-1 e MD5. O ICV aceita exclusivamente a assinatura assimétrica. O equipamento de origem encripta o controle dos dados do usuário e insere o resultado no campo de autenticação do ESP e o equipamento de destino confere se o dado foi alterado e se o endereço de origem é válido (GOLDANI 2004).

3.1.1 Modos do IPSec

O IPSec para atender as demandas entre tratar clientes e gateways trabalha em 2 diferentes modos tratando somente AH ou ESP onde ocorre a autenticação com o encapsulamento dos dados para o modo túnel no qual será desenvolvido nesta monografia.

Conforme Nakamura e Geus (2007) Modo de tunelamento – é geralmente utilizado pelos *gateways* IPSec, que manipulam o tráfego IP gerado por *hosts* que não aceitam o IPSec, como na modalidade que pode ser observada na figura 3.1. O *gateway* encapsula o pacote IP com a criptografia do IPSec, incluindo o cabeçalho de IP original. Ele, então, adiciona um novo cabeçalho IP no pacote de dados e o envia por meio da rede pública para o segundo *gateway,* no qual a informação é decifrada e enviada ao *host* do destinatário, em sua forma original.

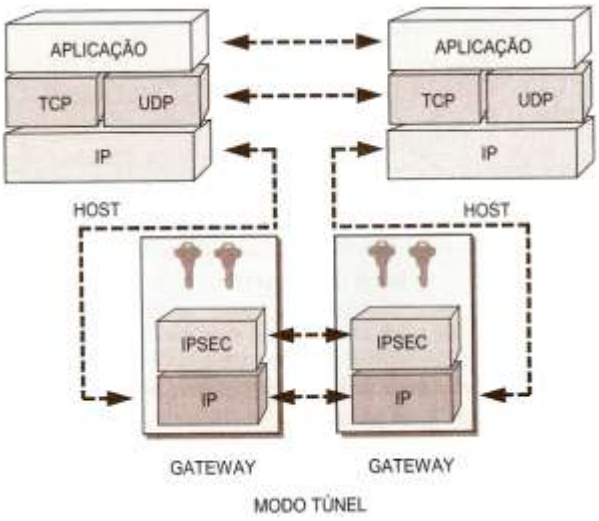

Figura 3.1 – No modo 'túnel', o IPSec é implementado no gateway
Fonte: Nakamura e Geus

Segundo Silva (2003), no modo túnel, todo pacote original e colocado dentro de um novo pacote, sendo gerados 2 novos cabeçalhos, 1 IP e outro ESP, bem como adciona os campos dados de autenticação e segmento de autenticação. A figura 3.2 mostra esse novo datagrama IP. Se o túnel for estabelecido entre dois servidores ou *hosts* os endereços de destino e origem do novo cabeçalho serão os mesmos daquele criptografado dentro do pacote. Se o túnel for feito entre gateways como roteadores ou firewall, os endereços de origem e destino do novo cabeçalho serão os dos gateways e os endereços dentro do pacote criptografados serão dos *hosts* ou servidores atrás do gateway.

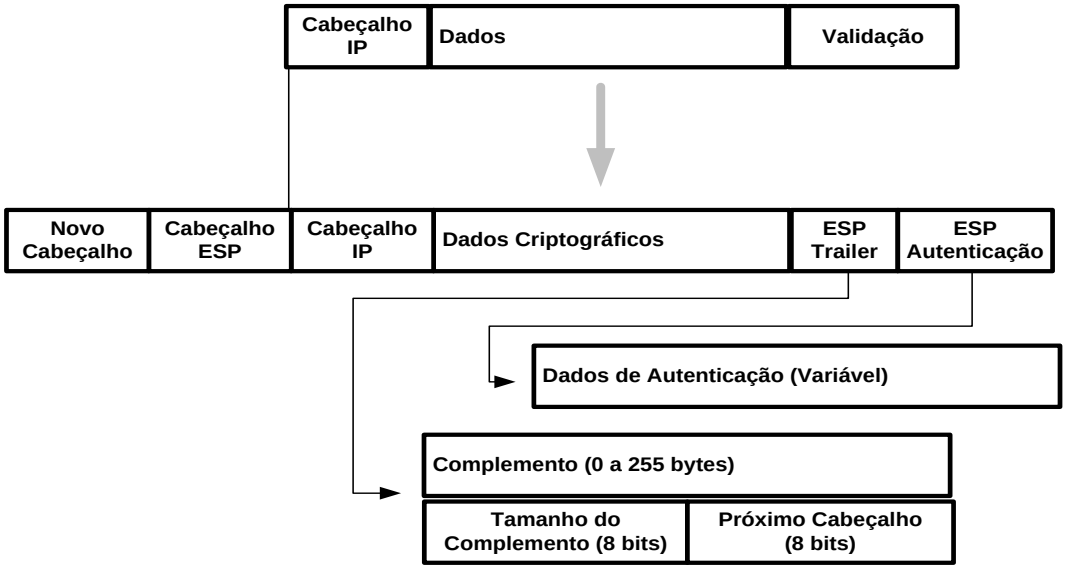

Figura 3.2 – Modo Túnel no protocolo ESP

Fonte: Silva 2003

3.1.2 Negociação de Chaves

Conforme Silva (2004) uma SA define os tipos de medidas de segurança que devem ser aplicadas aos pacotes baseados em quem está enviando os pacotes, para onde eles estão indo e que tipo de dados eles estão conduzindo. O conjunto de serviços de segurança oferecidos pela SA depende do protocolo de segurança, de suas opções escolhidas e do modo no qual a SA irá trabalhar.

Uma SA é identificada por três parâmetros: endereço IP de destino, identificação do protocolo de segurança (valor 51 para AH e o valor 50 para ESP) e o índice de parâmetro de segurança (*Security Parameter Index* - SPI). O SPI é um número que identifica uma SA, sendo definido durante a negociação que antecede o estabelecimento desta. Assim, todos os membros de uma SA devem conhecer o SPI correspondente e usá-lo durante a comunicação (Silva 2004).

Outro aspecto conforme Silva (2004) é que uma SA pode ser configurada manualmente por um administrador de segurança em cada *gateway* podendo ser

negociada dinamicamente por meio de um protocolo de gerência de chaves como o IKE - *Internet Key Exchange*.

Essa negociação dinâmica é necessária por várias razões: primeiro, porque não se sabe a priori quando será preciso negociar uma SA para estabelecer o túnel VPN e, segundo, porque uma associação de segurança não deve ter um tempo de vida infinito, ou seja, é recomendável que se troque a SA de tempos em tempos e, conseqüentemente, as chaves de criptografia. Quanto mais tempo se utilizar a mesma chave de criptografia, maiores serão as chances de algum invasor descobri-la.

O IKE é baseado no protocolo ISAKMP – Internet Security Association and Key Management Protocol, que prove um canal seguro entre dois pontos. O ISAKMP ocorre em duas etapas.

A primeira etapa é subdividida em dois modos de negociação:

a) Main Mode – Modo principal ocorre para o IKE estabelecer um canal seguro gerando um IKE SA para a segunda fase.

b) Aggressive Mode – Modo agressivo tem o mesmo principio do modo principal, mais simplificando, pois as identidades são tramitadas em conjunto com as solicitações de negociação, não fornecendo um canal seguro para que as informações sejam transmitidas.

A segunda etapa utiliza do canal providenciado na primeira etapa e faz a negociação de um novo túnel do SA para o IPSec, chamado de *Quick Mode* - modo rápido conforme a figura 3.3

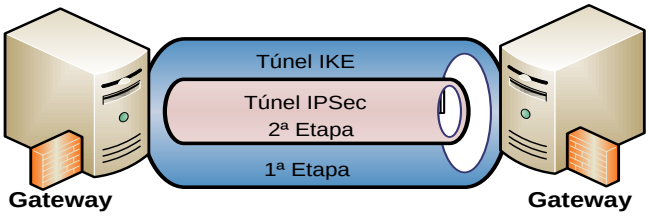

Figura 3.3 – Formação dos Túneis IKE / IPSec

Fonte: Autor 2008

Segundo Silva 2003 mesmo sendo extremamente difícil acontecer, suponha que um invasor consiga pegar a chave utilizada na primeira etapa da negociação. Esse intruso poderá descriptografar toda a comunicação da fase dois e derivar a chave da SA IPSec. A uma forma de se proteger nesse caso, usando uma técnica chamada *Perfect Forward Screcy*, na qual a chave AS IPSec será derivada por meio do algoritmo *Diffie Hellman*.

A geração de chaves do protocolo IKE é feita por meio do algoritmo *Diffie Hellman.* Esse algoritmo não tem por objetivo criptografar os dados e simprover uma maneira rápida e eficiente de troca de chaves criptográfica, entre dois sistemas, baseadas nas duas parttes de chave (pública e privada) de cada interlocutor.

Algumas aplicações podem ser verificadas com a utilização do IPSec, nesta monografia iremos trabalhar com a VPN, conforme a seguir no tópico 3.2.

3.2 Utilização de VPN

Conforme Nakamura e Geus (2007) as redes virtuais privadas tem uma importância fundamental para as organizações, principalmente no seu aspecto econômico, ao permitirem que as conexões dedicadas sejam substituídas pelas conexões públicas. Além do que ocorre com as conexões privadas, também é possível obter economia com a substituição das estruturas de conexões remotas, que podem ser eliminadas em função da utilização dos clientes e provedores VPN**.**

Neste contexto de VPN observam-se implicações para o canal de comunicação:

a) Desempenho em relação à conectividade.

b) Aumento da complexidade das conexões.

c) Aumento do número de conexões que devem ser gerenciados.

d) Aumento dos custos conforme o aumento do número de integrantes do ambiente.

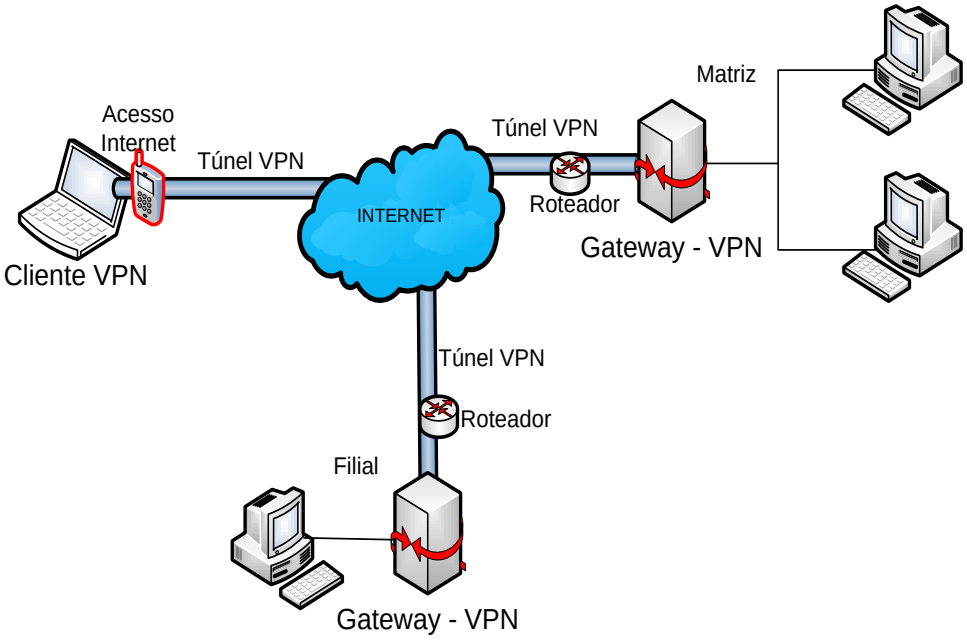

Figura 3.4 - Demonstrativo de possíveis VPNs

Fonte: Autor 2008

Conforme a figura 3.4 observar-se que a comunicação VPN deriva tanto de um gateway para outro gateway como também de uma estação cliente remota para um gateway. Os usuários remotos fazem uma conexão discada para um provedor de acesso à Internet é possível estabelecer uma VPN entre os gateways da VPN que protegem uma filial. Cabe ao gateway estabelecer a permissão ou negação dos usuários remotos ou de uma filial para dentro da rede protegida por esse gateway.

Podem-se aplicar diversos tipos de topologias, conexões entre filiais, empresas distintas ou usuários remotos simultaneamente. O quantitativo de conexões indiferente para criação das VPNs o que pode ocorrer o aumento de gerenciamento no processo. No decorrer desse capítulo verificam-se os modos túnel e transporte para criação de VPN.

Nesse contexto, as VPN aparecem para superar o problema de segurança. Usando protocolos de tunelamento e procedimentos de ciframento, a integridade e autenticidade dos dados são garantidas. Como as operações ocorrem sobre uma

rede pública, a implementação e manutenção de uma VPN custa significativamente menos do que os serviços dedicados descritos no primeiro parágrafo.

Conforme Cheswick (2003), uma VPN poderia ser definida como um conjunto de computadores protegidos da Internet via um firewall, se eles estivessem realmente conectados. Essas máquinas eram mais seguras contra ataques externos, porque o dinheiro, a perícia e a paranóia estavam todos concentrados na manutenção do gateway seguro. Dessa forma, os sites com múltiplas localizações tinham de ser interligados de forma privada, uma vez que a Internet não oferecia serviços suficientemente seguros para interligar localizações. Cada site era protegido por um firewall, mas não havia maneira alguma para máquinas em diferentes locais se comunicarem com segurança. Devido ao firewall, era improvável que elas pudessem se comunicar de qualquer maneira.

As redes privadas virtuais ampliam os limites de um domínio protegido por meio do criptografia. Há três tipos de VPNs. O primeiro permite a filiais remotas compartilhar um perímetro de segurança e até mesmo um espaço de endereçamento. O segundo é utilizado porque não querem abrir suas redes inteiras umas para as outras, mas desejam ter alguns serviços dos compartilhados - essas VPNs implementam uma DMZ. O terceiro tipo permite ao remotos se conectar a seu local de trabalho a partir de casa, do hotel ou café-bar (CHESWICK 2003).

Um problema enfrentado na VPN pelas empresas que tem acesso direto à Internet está na quantidade de endereços IPs que ele pode disponibilizar para seus usuários remotos. Cada vez mais o crescimento da *internet* produz escassez de endereços IPv4. Sendo assim ocorre a necessidade de prover um método no qual os endereços podem ser traduzidos para ser representados um a um ou de um conjunto de endereços por um único endereço. No caso de uma pequena empresa tentando compartilhar o modem e a linha telefônica, o problema é um pouco mais complicado; nesse caso o único meio de acesso à Internet é uma conta em um provedor de acesso. Essas contas de acesso dão direito a apenas um endereço IP, que o computador recebe quando disca para o provedor, neste caso, este endereço IP é alterado a cada conexão que for feita, assim sendo, será necessário fazer com que todas as máquinas locais compartilhem este endereço IP.

Estes IPs são endereços de Classe A, B ou C. Para resolver esses problemas, a saída se baseia numa técnica chamada *Network Address Translation* (NAT), ou Tradução de Endereço de Rede. O NAT é apenas uma série de tarefas que um roteador (ou equipamento equivalente) deve realizar para converter endereços entre redes distintas (SILVA). Conforme as figuras abaixo se exemplificam a formação de NAT *STATIC* e NAT *HIDE.*

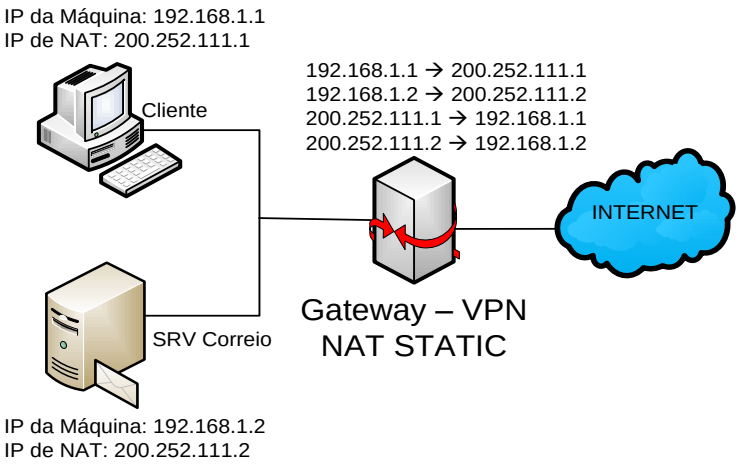

Figura 3.5 – NAT *STATIC*

Fonte: Autor 2008

Pode-se observar na figura 3.5 a formação da NAT *STATIC* fazendo a tradução de endereços um a um, dos privados para os públicos.

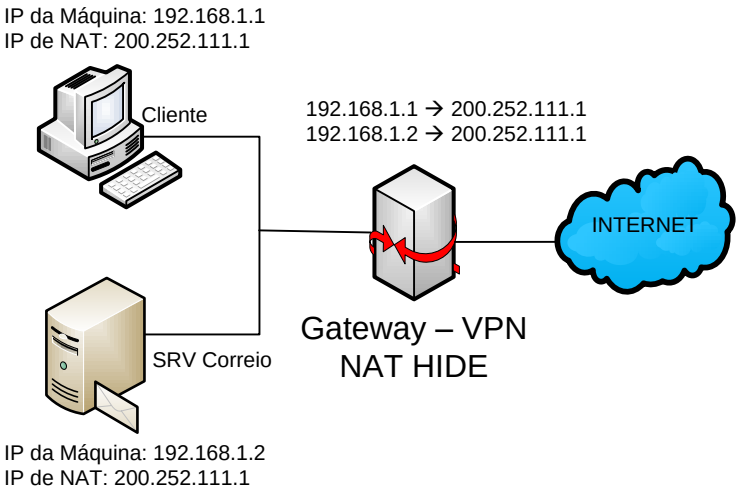

Figura – 3.6 NAT *HIDE*

Fonte: Autor 2008

Observa-se na figura 3.6 a formação de NAT HIDE, na qual, permite tradução de vários endereços privados para somente um único endereço publico.

Verificou-se nesse capítulo como são utilizados os modelos criptográficos para proteção do cabeçalho IP e dos dados a esse atrelado. O conjunto de padrões IPSec traz modificações no modo de transmissão do pacote, entre sua origem até seu destino e a VPN utiliza do IPSec para prover um túnel seguro. No decorrer dessa monografia faz-se entender o funcionamento na prática do IPSec e será apresentado a captura do pacote e o comparativo de seu desempenho.

3.3 Mecanismos de defesa IPSec

Conforme mencionado no capitulo 2 as possibilidades de ataques ao protocolo TCP/IP apresenta-se neste tópico os mecanismos de defesa no qual a utilização do IPSec poderá suprir os problemas existentes.

3.3.1 Defesa Packet Sniffing

A técnica de defesa para ataques packet sniffing consiste na captura de informações diretamente pelo fluxo de pacotes que trafegam no mesmo seguimento da rede em que o software *sniffer* funciona, sendo assim o IPsec prove confidencialidade do fluxo de trafego através do tunelamento dos gateways não permitindo que um software *sniffer* faça a captura da informação trafegada por esse túnel.

3.3.2 Defesa Captura de Usuário e Senha nas seções FTP e Telnet

As sessões FTP e Telnet trafegam com seus usuários e senhas em claro permitindo com que um hacker identifique o acesso e posteriormente façam uso dos meios de autenticação para se passar pelo usuário. Se for feita uma autenticação de dados identificando a fonte original do trafego o hacker fica impossibilitado de ter

acesso as informações ao menos que ele quebre a senha. Mesmo que ele consiga quebrar a senha as informações que passam através do túnel IPsec estão cifradas e ele não ira conseguir identificar o conteúdo da informação original.

3.3.3 Defesa TCP SYN Flood Ataque

O TCP SYN ataque tira vantagem do comportamento de 3 *WAY HANDSHAKE* efetuado pelo protocolo TCP no processo de uma conexão. O atacante faz um pedido de conexão para o servidor de vitima com pacotes que carregam o endereço falsificado do IP de origem. Como resultado o servidor da vitima perde tempo e recursos de máquina. Através do uso de integridade sem conexão onde é possível usar IPsec para verificar a integridade de qualquer pacote IP individual sem precisar referenciar qualquer outro pacote; cada pacote pode permanecer sozinho ou ser validado sobre si próprio. Sendo assim no 3 way handshake o atacante dependeria que os vários pedidos de conexão fossem aceitos simultaneamente o que não ocorre nesse sistema, mesmo se isso ocorrer os IPsec possui um mecanismo de defesa contra ataque de pacotes repetidos que é um contador de pacotes que protege contra esses tipos de ataques.

3.3.4 Defesa TCP *Session Hijacking* TCP

Session hijacking também conhecido como ataque *man in the middle,* é possivel quando um *hacker* toma controle de uma sessão TCP entre duas maquinas. Normalmente a autenticação ocorre somente no início de uma sessão TCP. Com um mecanismo de controle de acesso e autenticação pode-se evitar ataques desse tipo, mesmo que eles ocorram com sucesso, a informação estará cifrada não permitindo ao atacante o domínio dos dados originais.

3.3.5 Defesa IP Spoofing

O IP *Spofing* é um método de ataque que consiste na criação do pacote IP utilizando endereço IP de outra origem. Novamente um mecanismo de autenticação e controle de acesso pode prevenir esses tipos de ataques mesmo que eles ocorram existe um túnel IPsec onde a informação é cifrada e com um fluxo de trafego limitado.

Conforme verificou-se neste capítulo pode-se identificar a importância da criação de uma VPN com a utilização do conjunto de padrões IPSec para promover diversos tipos de proteção contra ataques ao TCP/IP e garantido sigilo da informação tramitada.

4 PROPOSTA DE SOLUÇÃO E MODELO

Neste capítulo apresenta-se um modelo de segurança da informação, com aplicabilidade do conjunto de padrões IPSec, e ferramentas utilizadas para captura do trafego em claro e criptografado IPSecmon , na qual, posteriormente será descrito a análise de desempenho.

4.1 Apresentação Geral do Modelo Proposto

Para que possamos prover devida segurança na tramitação dos dados entre uma estação e outra deve-se ter em mente a garantia da confidencialidade e integridade das informações a serem trafegadas.

Apresenta-se na figura 4.1 a visão geral da topologia utilizada para simular o tráfego das informações em claro e criptografadas. Demonstra-se também um método de ataque no qual chamamos de *sniffing,* ferramenta utilizada para capturar os dados transitados no meio, na qual, a mesma é utilizada para identificar o tipo de pacote que será transportado.

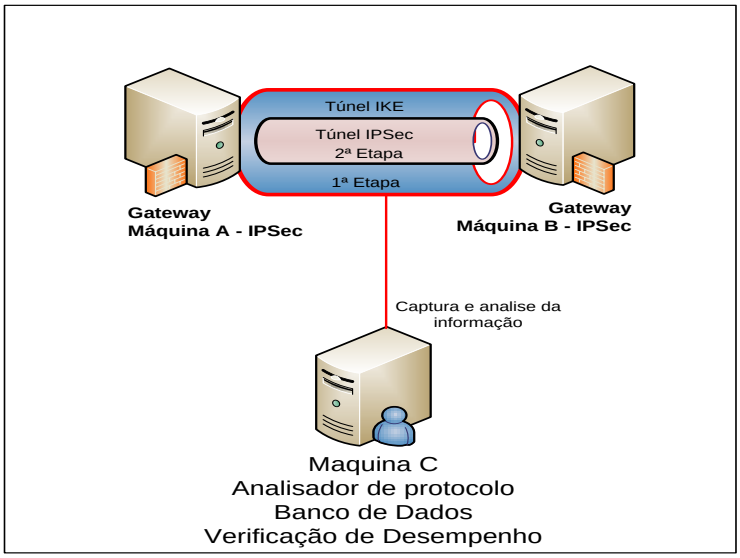

Figura 4.1 – Topologia Geral

Fonte: Autor 2008

Conforme demonstrado na figura 4.1 será utilizado uma VMware que é um virtualizador de máquinas onde pode-se operar com um SO - sistema operacional dentro de outro, com a instalação de três outros sistemas operacionais sendo:

a) Dois SO software livres Debian Etch versão 2.6.18 para simular a tramitação em claro e a tramitação criptografada com o IPSec.

b) Um SO Windows XP ou Ubuntu com a utilização de um analisador de protocolo para capturar os dados e identificar o tipo do pacote. Na primeira fase será utilizado o *wireshark,* já na segunda fase será desenvolvido um analisador de protocolos desenvolvido em linguagem de programação C com a biblioteca libcap que armazenará as informações em um banco de dados em software livre MySQL para, posteriormente, serem tratadas as informações capturadas e analisar o desempenho do método utilizado IPSec. A extração das informações e apresentação das mesmas será desenvolvida com a linguagem PHP.

4.2 Metodologia Aplicada

A metodologia aplicada constitui no desenvolvimento de uma ferramenta de captura de pacotes, IPSecMON, armazenamento das informações coletadas em um banco de dados. Posteriormente será providenciado uma análise de desempenho comparativa dos dados trafegados em claro e dos dados trafegados cifrados com a utilização do túnel VPN com o IPSec, no qual, serão demonstrados em página com a utilização do jpgrafic.

4.2.1 Criação do túnel VPN

Para que os dados possam ser tramitados de um ponto ao outro com a devida segurança é necessário avaliar os aspectos dos protocolos existentes.

4.2.1.1 Problemas do tráfego em claro

O tráfego baseado em IPv4 não oferece segurança. Todo o texto trafega em claro, podendo o atacante capturar as informações e identificá-las. O atacante também tem a opção de diversos tipos de ataques para neutralizar o sistema e até mesmo interrompe-lo dependendo do tipo de ação. Alguns destes ataques conforme apresentados no capítulo 2 podem ser mencionados:

a) IP Spoofing.

b) SYN Flooding.

c) TCP Session Hijacking.

d) Proteção de Usuário e Senha nas Sessões de FTP e Telnet.

4.2.1.2 Proposta de Segurança com VPN

Para a aplicabilidade de segurança no texto trafegado em claro é utilizado o método de atribuição de uma VPN para trafegar a informação entre a máquina A (Alfa) e B (Beta). As máquinas de comunicação estarão com o sistema operacional Linux Debian Etch versão 2.6 instalada. Esse modelo prove segurança com a utilização do conjunto de padrões IPSec em relação ao IPv4.

O modelo criptográfico utilizado pelo IPSec na transmissão de pacotes prove autenticação e confidencialidade. O IPSec resolve muitos dos ataques e das falhas de segurança ao protocolo TCP/IP.

Ficou definido a utilização do IPSec ESP ao invés do AH, pois alem de fazer a autenticação do cabeçalho o modo ESP encapsula o *payload* (massa de dados). Em relação de utilização do modo túnel ou transporte no IPSec definiu-se o modo túnel pois neste podemos trafegar de gateway para gateway conforme apresentado no capítulo 3.

Posteriormente será utilizado em uma máquina Windows XP a ferramenta de captura de pacotes *Wireshark* para poder identificar o texto trafegando em claro e criptografado.

A criptografia é efetua na camada IP do modelo de referência TCP/IP e sendo assim esta não adiciona um nível de segurança diretamente as aplicações que são executadas pelo usuário. O IPSec pode ser suscetível a falhas dependendo de sua implementação ou pelas vulnerabilidades dos tipos dos métodos criptográficos utilizados.

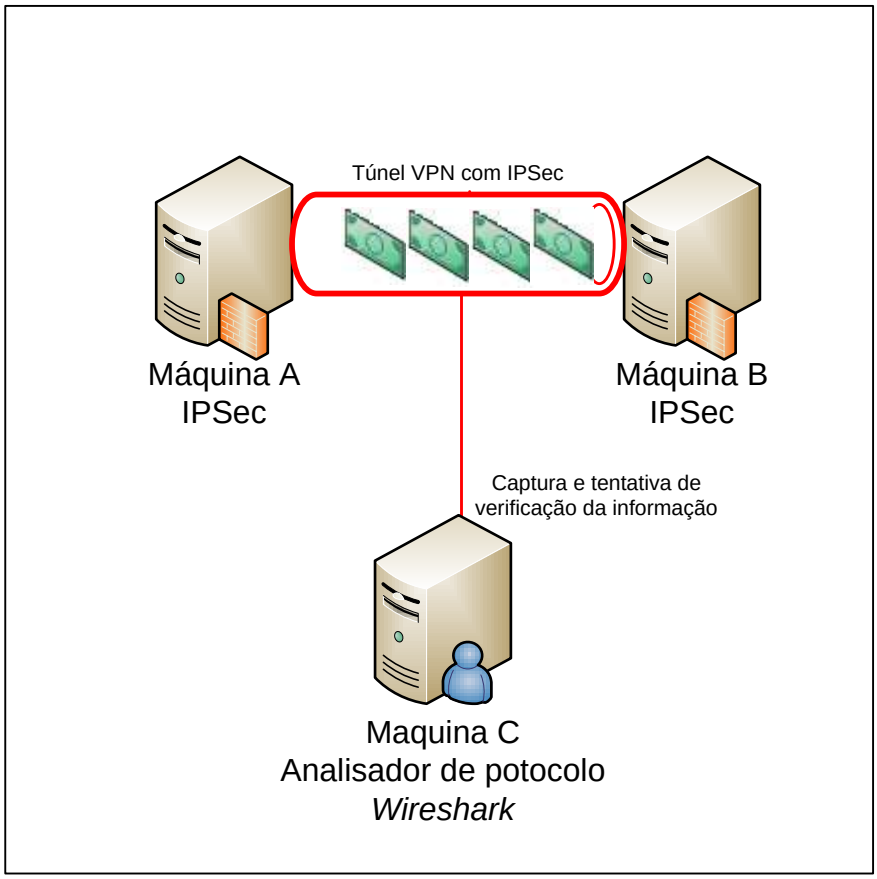

Figura 4.2 Topologia de Implementação VPN

Fonte: Autor 2008

4.2.1.3 Implementação de VPN

A plataforma utilizada foi o Debian Etch, versão 2.6.18, no qual, foi realizada a instalação básica do Debian.

No primeiro acesso ao Debian, foi realizada a atualização dos pacotes com o seguinte comando:

➔ **apt-get update**

Para prover o serviço de VPN foi utilizada a ferramenta OPENSWAN nas duas máquinas da arquitetura:

➔ **apt-get install openswan**

Foi determinado o endereço IP das duas maquinas via DHCP:

➔ **ifconfig**

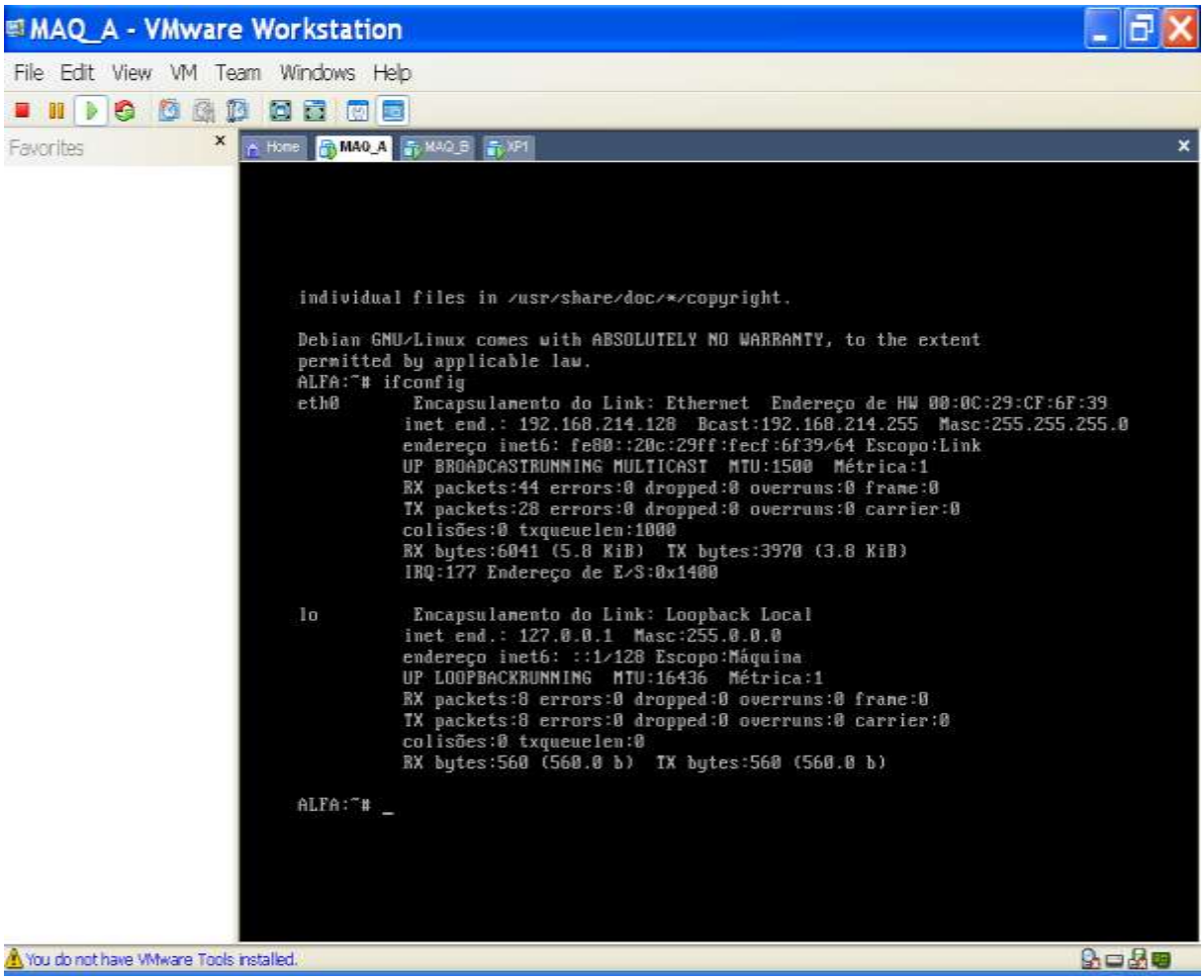

Figura 4.3 Endereço IP DHCP

Fonte: Autor 2008

Estes endereços IP são recebidos via DHCP nas máquinas e serão usados na configuração do IPSec para definir os gateways IPSec.

O arquivo principal de configuração é o */etc/ipsec.conf* , no qual, foi editado com os seguintes dados em ambas as máquinas:

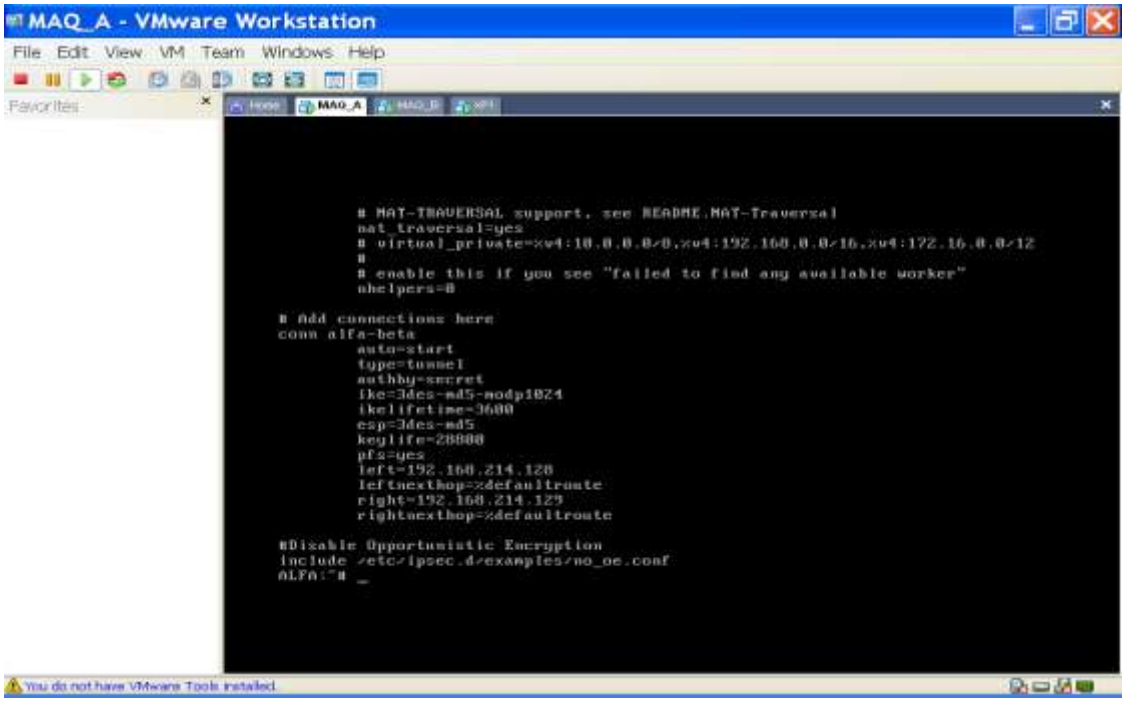

Figura 4.4 Configuração do arquivo /etc/ipsec.conf

Fonte: Autor 2008

Outro arquivo importante que deve ser configurado para que a conexão seja habilitada automaticamente, tipo túnel e com autenticação por senha compartilhada é o arquivo */etc/ipsec.secret,*onde encontra-se definida a senha que será usada na negociação. Este arquivo encontra-se nas duas máquinas.

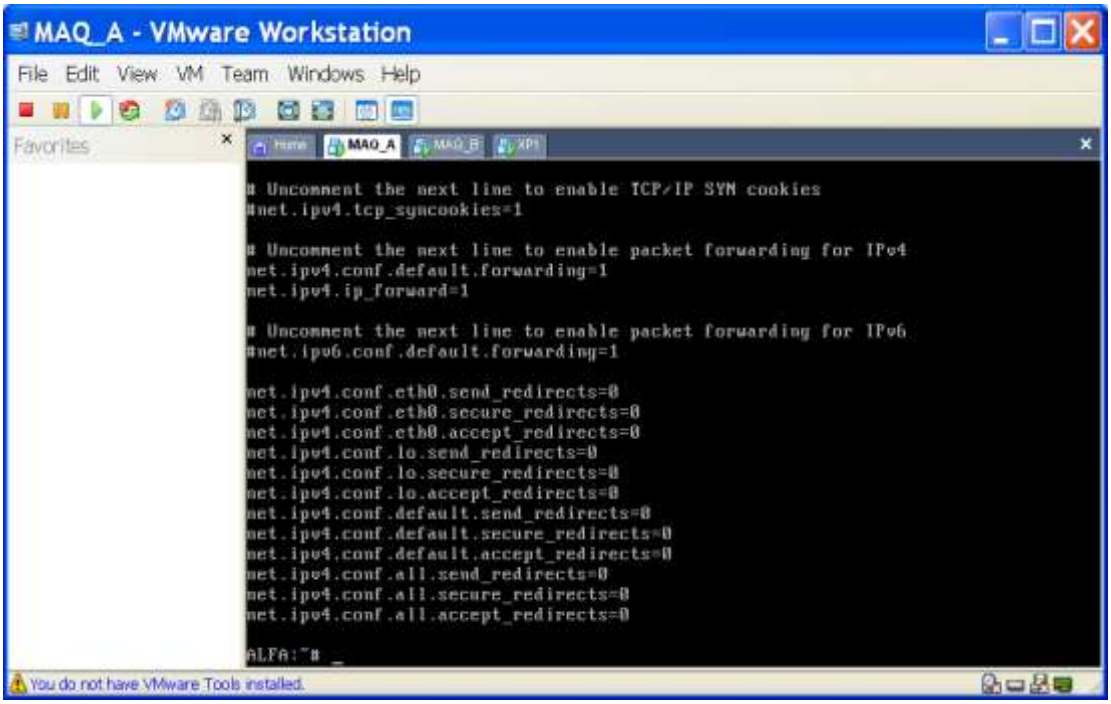

Figura 4.5 Arquivo de configuração de Senha IPSec

Fonte: Autor 2008

Deve-se habilitar o roteamento (IP_FORWARD) e desabilitar o redirect de pacotes ICMP nas duas máquinas. Para tal foi editado o arquivo */etc/sysctl.conf* nas duas máquinas.

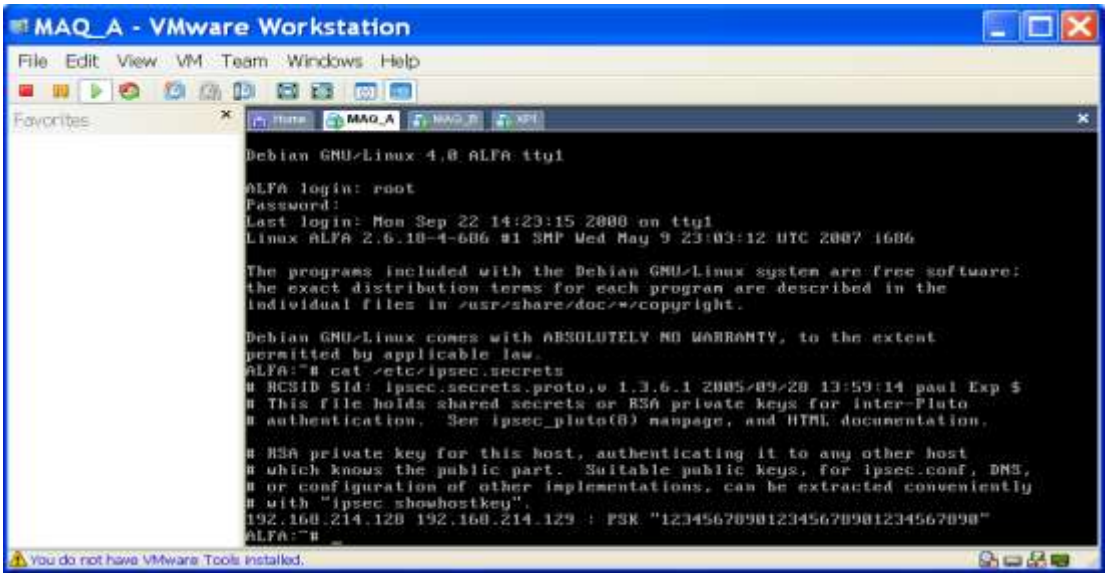

Figura 4.6 Habilitar o Roteamento

Fonte: Autor 2008

Para iniciar o serviço do IPSEC foi executado o seguinte comando:

➔ /etc/init.d/ipsec start :

Figura 4.7 Inicializando o IPSec

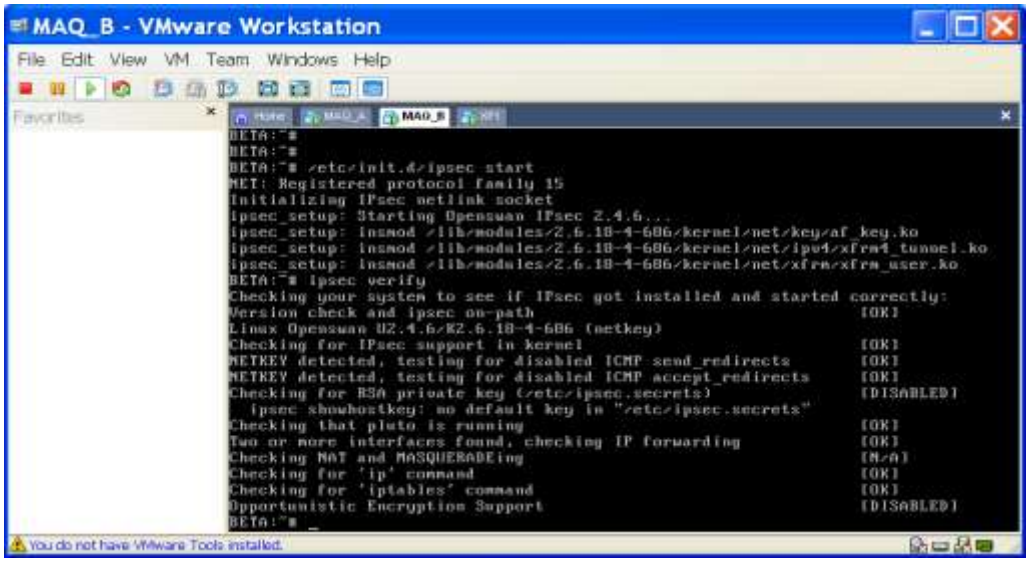

Figura 4.7 Inicializando o IPSec

Fonte: Autor 2008

Para verificar o funcionamento do IPSec pode-se utilizar o comando:

➔ **ipsec verify**

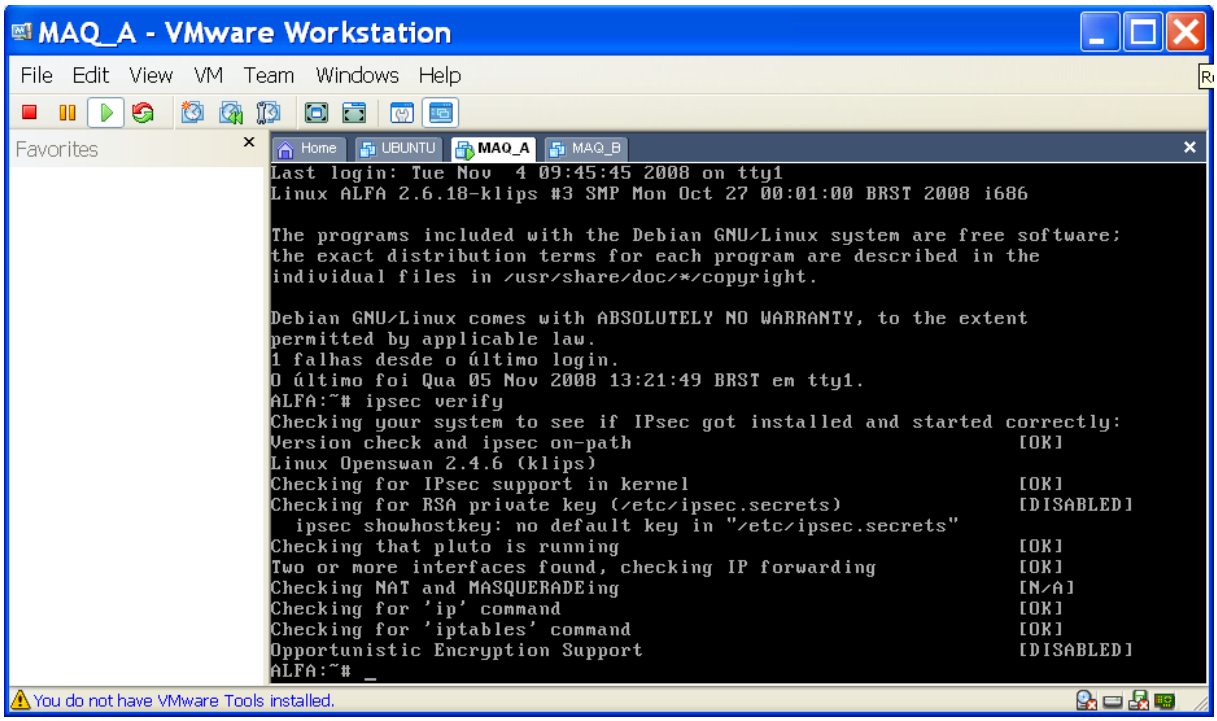

Figura 4.8 Verificação de Funcionamento IPSec

Fonte: Autor 2008

Após verificar o funcionamento do IPSec nas duas máquinas inicia-se o processo de captura dos pacotes em claro e criptografados com o analisador de protocolos *WireShark versão 1.0.3*. Pode-se verificar na figura 4.9 a negociação do protocolo ISAKMP, utilizando o UDP na porta 500, entre os endereços de origem e destino fechando o túnel no *main mode*, entre as máquinas A e B. a linha de comando para a filtragem de pacotes no campo *Filter* utilizada foi:

➔ (ip.src==192.168.214.128 and ip.dst==192.168.214.129) or
(ip.src==192.168.214.129 and ip.dst==192.168.214.128)

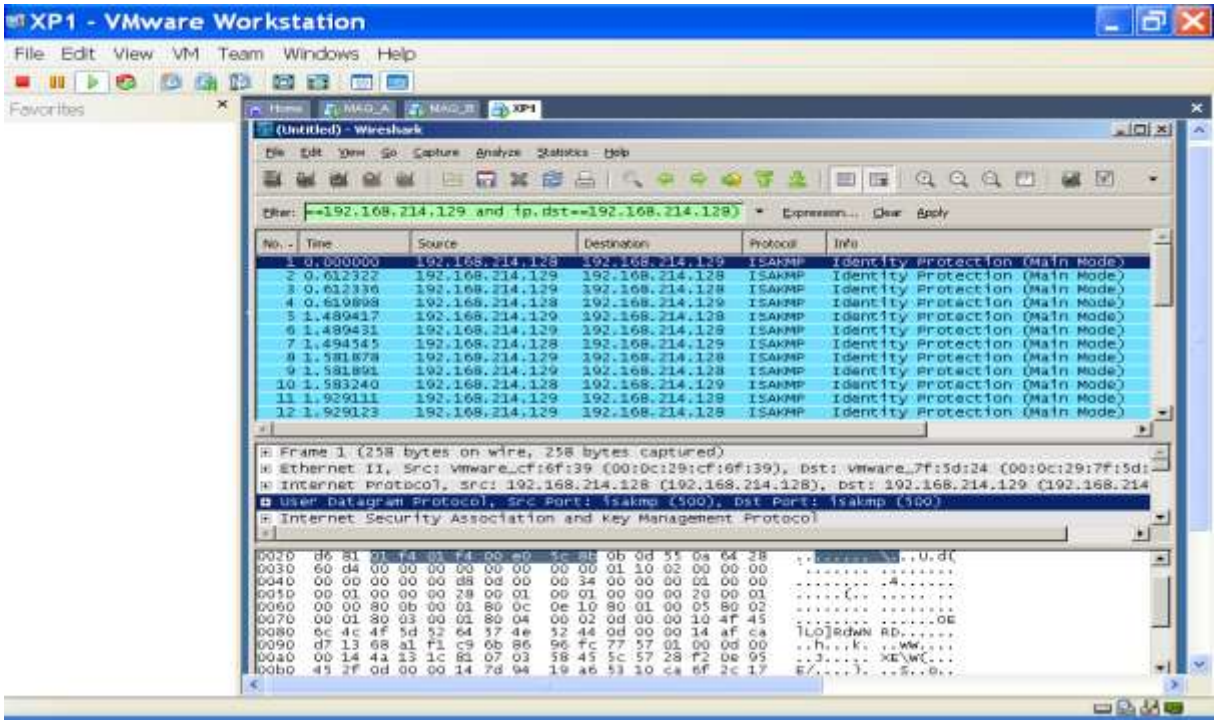

Figura 4.9 Captura *WireShark Main Mode*

Fonte: Autor 2008

Após ocorrer a negociação do *Main Mode* identifica-se na figura 4.10 a negociação do *Quick Mode* no decorrer da filtragem.

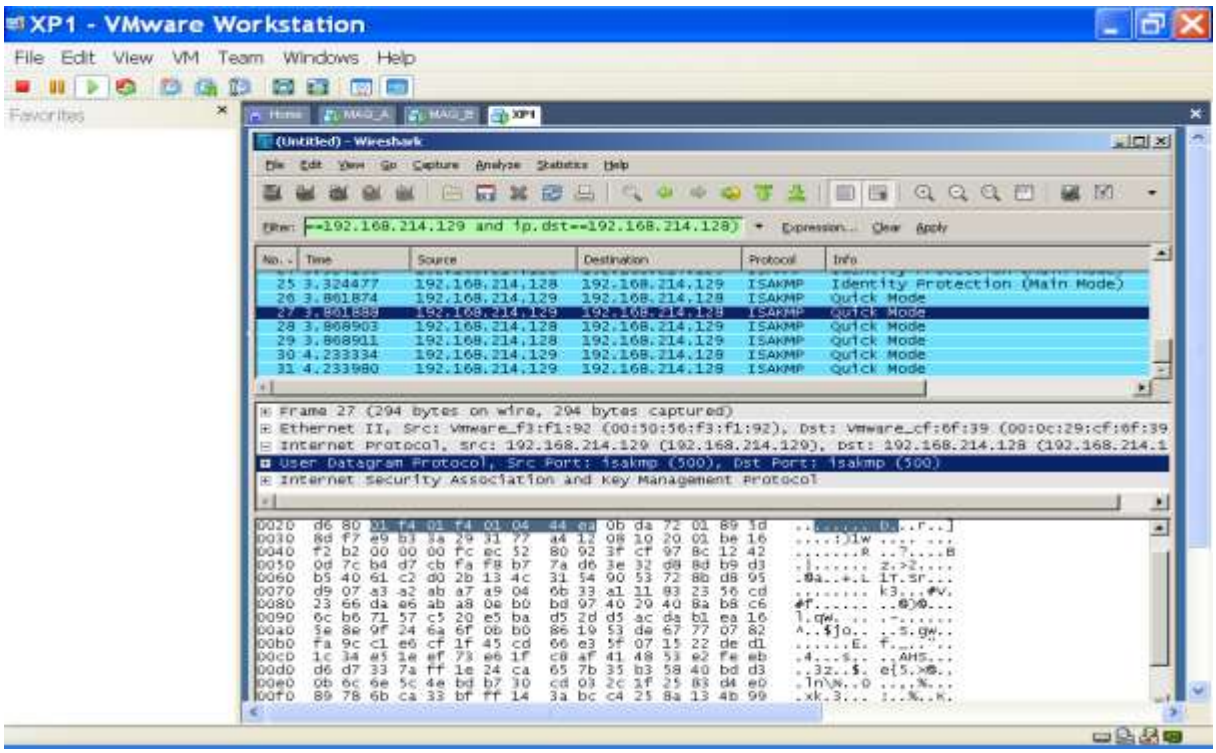

Figura 4.10 Captura WireShark Quick Mode

Fonte: Autor 2008

Nesse momento o ipsec será desativado com o comando:

➔ /etc/init.d/ipsec stop

Pode-se identificar a comunicação entre a máquina A e B sem criptografia com a utilização do comando ping pacote ICMP conforme a figura 4.11.

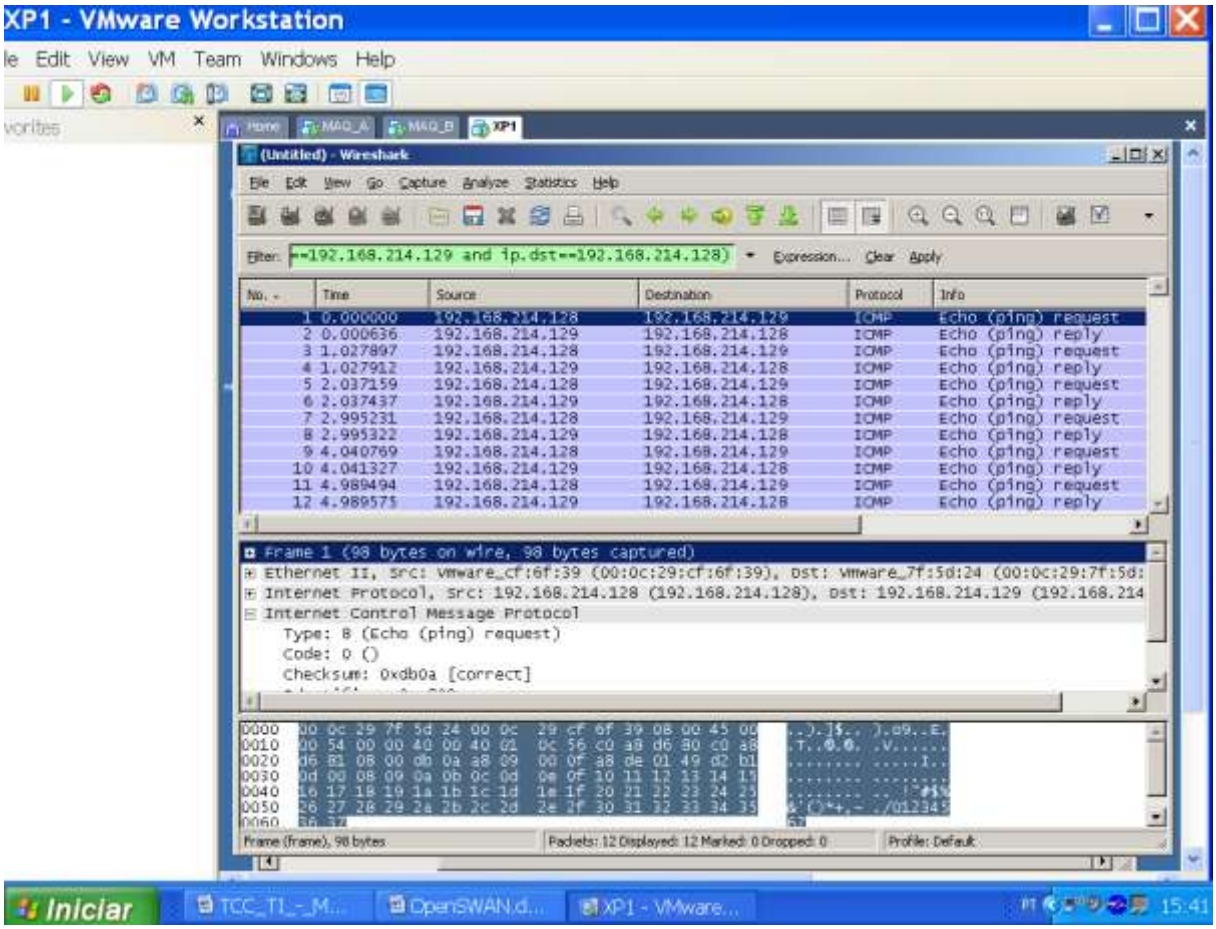

Figura 4.11 Captura *WireShark* sem Criptografia

Fonte: Autor 2008

Conforme figura 4.11, observa-se após a inicialização do IPSec que o mesmo comando ping foi utilizado, mas não identifica-se o pacote ICMP e somente verifica-se o protocolo ESP, caracterizando a criptografia do cabeçalho e dos dados trafegados.

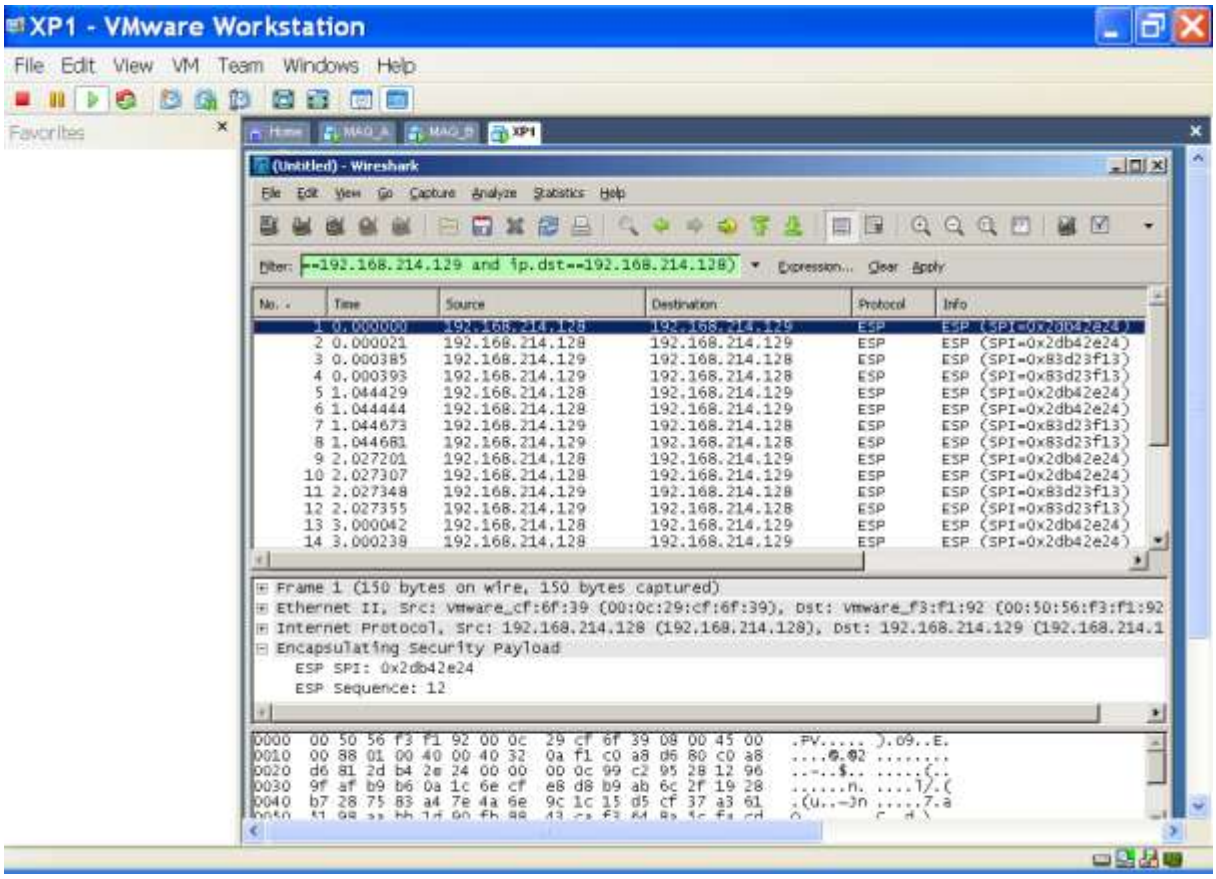

Figura 4.12 Captura WireShark Protocolo ESP

Fonte: Autor 2008

Conforme apresentado na criação de VPN, identifica-se que com o protocolo IPSec os dados apresentam-se criptografados provendo devida segurança na tramitação dos dados.

Com base no *WireShark*, será implementada uma ferramenta de análise de protocolos para capturar, identificar os pacotes tramitados e fazer uma avaliação quanto ao desempenho em relação a criptografia inserida no processo de transmissão dos dados em claro.

4.2.2 *Criação do Capturador e Analsiador de Protocolos*

Após a identificação da arquitetura do projeto em funcionamento para criação da VPN, com o IPSec habilitado, será utilizado uma ferramenta de captura dos

dados trafegados, enviá-los para o banco de dados e posteriormente poder fazer uma análise de desempenho dos dados em claro e cifrados.

4.2.2.1 Problema de Captura de Pacotes

Um problema identificado em relação ao programa *Wireshark,* que apesar de fazer uso de uma biblioteca de captura de pacotes chamada *libpcap* e ainda possuir instalados analisadores de protocolos de rede, exige um armazenamento das informações obtidas da análise dos pacotes trafegados, em especial o protocolo IPSec para levantar estatísticas de conexões da tarefa executada. Com o *Wireshark* é possível realizar uma captura e analisar os protocolos envolvidos nesta captura e gravar no disco rígido o resultado deste trabalho (arquivo pcap), porém seria necessária uma intervenção humana no processo para fazer o parser desse arquivo e carregá-lo num banco de dados, para posterior análise. Esse processo não ocorre em tempo real e de forma automatizada.

4.2.2.2 Proposta de desenvolvimento IPSecMON

A sugestão de resolução do problema supracitado envolve o desenvolvimento de um programa, no qual, será batizado como IPSecMON, com a utilização do sistema operacional Debian Etch 2.6.18 configurado com o Klips em seu kernel que em parte utiliza de algumas tecnologias presentes no programa *Wireshark*.

O programa IPSecMON em questão fará uso da linguagem de desenvolvimento C e da biblioteca libpcap mais a biblioteca libMySQL 1.5 dev, para capturar os pacotes de rede que trafegam entre os gateways VPN e transmitir a informação capturada, este analisador de pacotes específico identificará os dados antes de serem criptografados, considerados em claro, e após serem criptografados no protocolo IPSec. Estes dados capturados serão armazenados num banco de dados MySQL. Esse capturador de pacotes IPSec rodará como um serviço de agente nas máquinas Gateway.

Uma segunda parte nesta, envolve a análise e visualização destes dados, para tanto será usado um servidor apache com o *engine* PHP instalado. O PHP fará

o acesso ao banco de dados MySQL e mostrará a estatística de acesso em tempo real. Posteriormente será utilizada a biblioteca JpGraph para visualizar gráficos destas estatísticas.

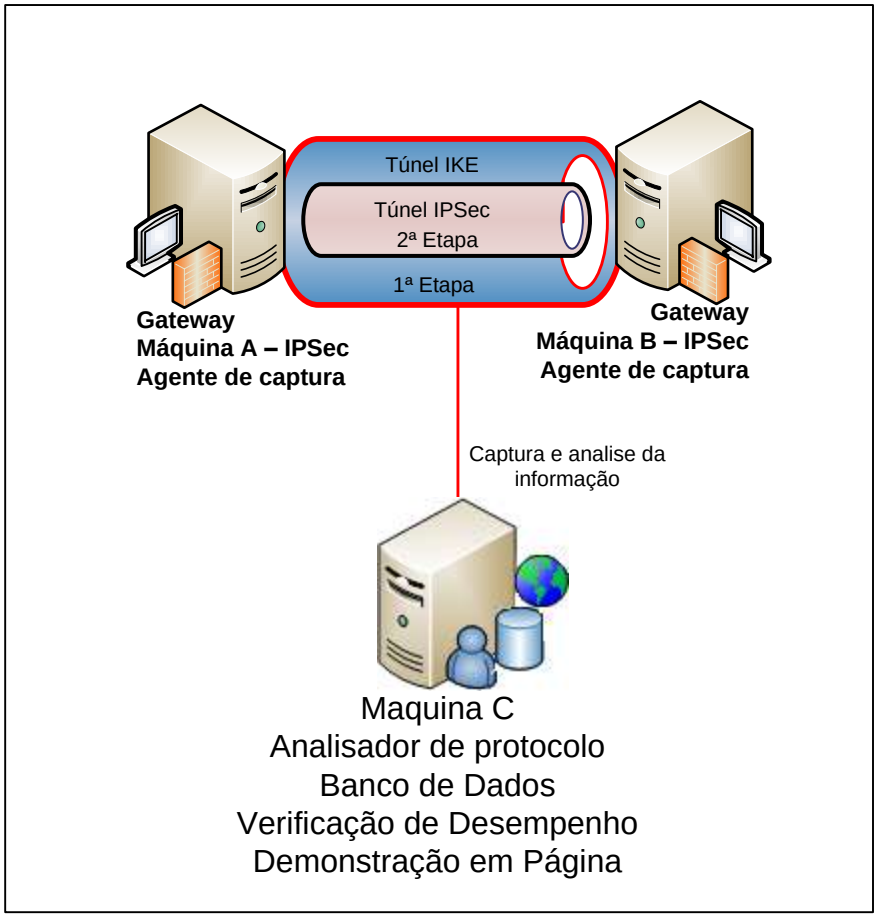

Figura 4.13 Topologia de Implementação IPSecMON

Fonte: Autor 2008

4.2.2.3 Implementação IPSecMON

Para que possamos capturar e medir o desempenho entre o tráfego em claro e cifrado, foi desenvolvido e configurado tanto na máquina Alfa – A quanto na máquina Beta – B o agente IPSecMon.

O IPSecMon é um agente que trabalha com a biblioteca libcap para fazer captura dos pacotes trafegados e posteriormente encaminhá-las para o banco de dados MySQL.

O arquivo de configuração do agente IPSecMon é o ipsecmon.conf e encontra-se no diretório /etc. Neste arquivo foram configurados os seguintes comandos:

Conforme figura 4.14 apresenta-se o direcionamento de onde os arquivos *daemon* devem se encontrar.

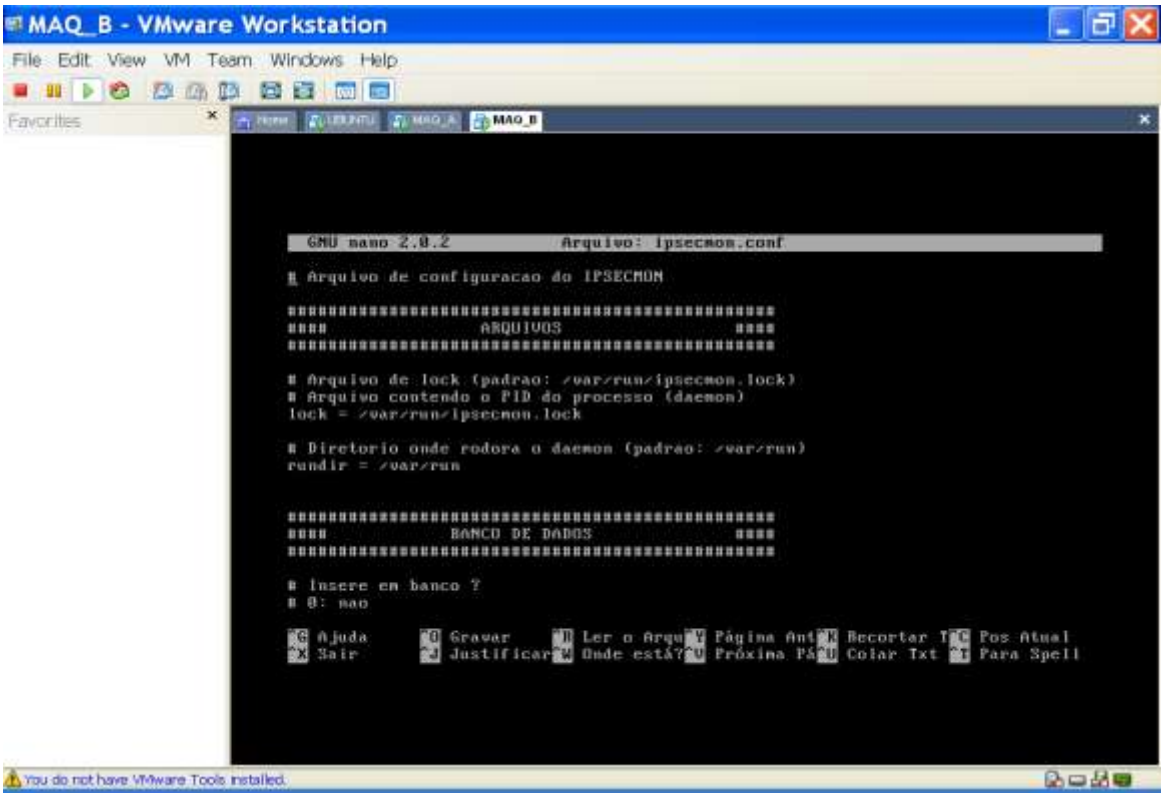

Figura 4.14 IPSecmon.conf Arquivos

Fonte: Autor 2008

Conforme figura 4.15 demonstra-se onde devemos incluir os tipos de cenários para captura dos tamanhos de arquivos tramitados tanto para os textos em claro, quanto cifrados, a serem executados e armazenados no banco de dados as informações. A descrição Zerar indica eliminar toda a informação no banco de dados. Posteriormente será utilizado a descrição em tamanho de MB. É atribuído 010MB, 050MB, 100MB, 150MB e 250MB para os dados com o IPSec desabilitado, texto em claro e as descrições 010MB, 050MB, 100MB, 200MB e 400MB para os dados com o IPSec habilitado, texto cifrado.

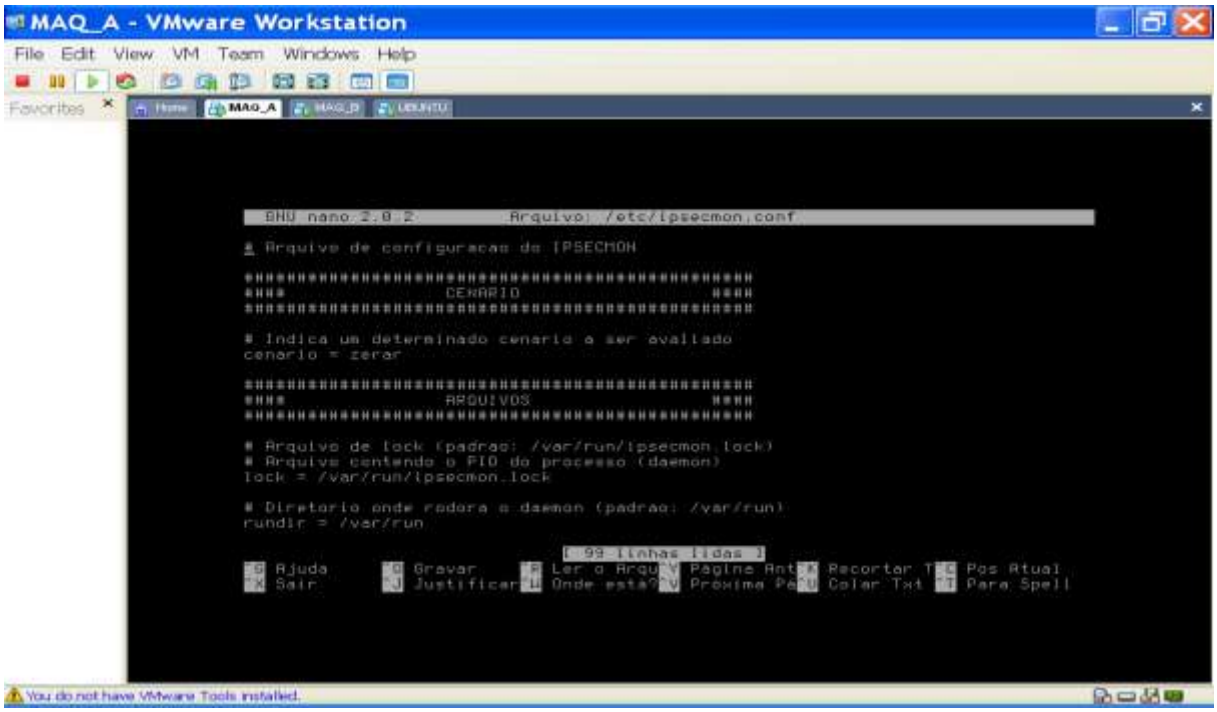

Figura 4.15 IPSecmon.conf Cenário

Fonte: Autor 2008

Conforme figura 4.16 apresenta-se a habilitação, usuário, senha e endereço do banco de dados.

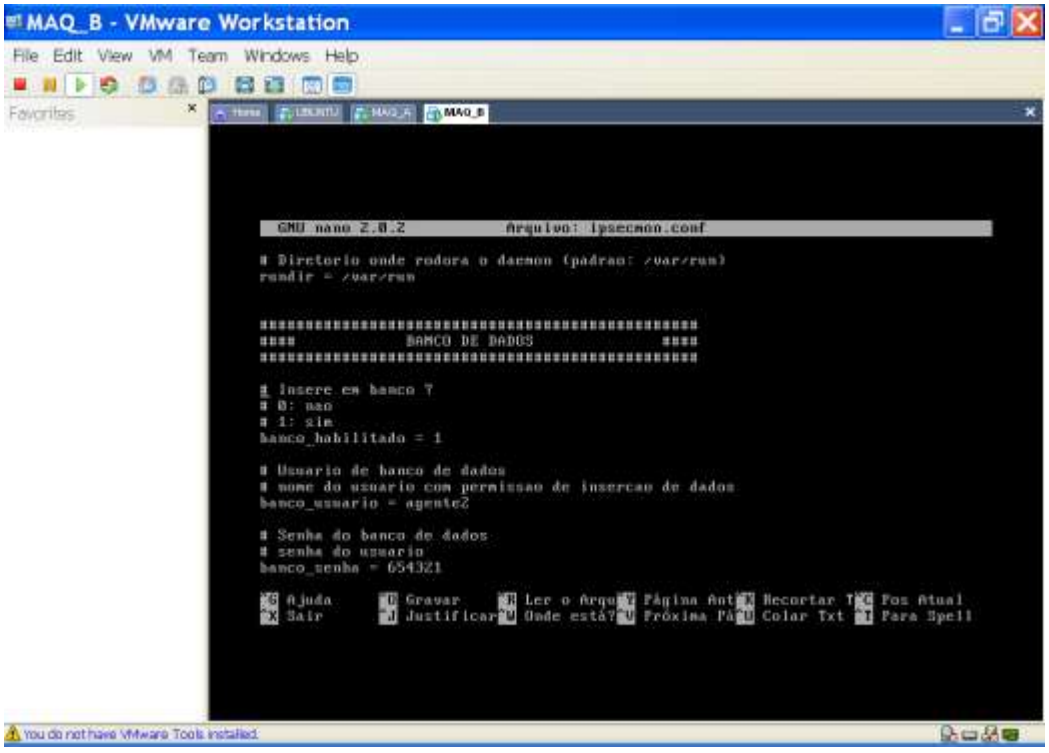

Figura 4.16 IPSecmon.conf Banco de Dados

Fonte: Autor 2008

Conforme figura 4.17 apresenta-se a captura da eth0, a não utilização do modo *promisc* e o uso do filtro de captura para os endereços das máquinas A e B. Na interface dos agentes é configurado a captura dos textos trafegados em claro com a interface1 = etho e dos textos trafegados criptografados com a interface1 = ipsec0.

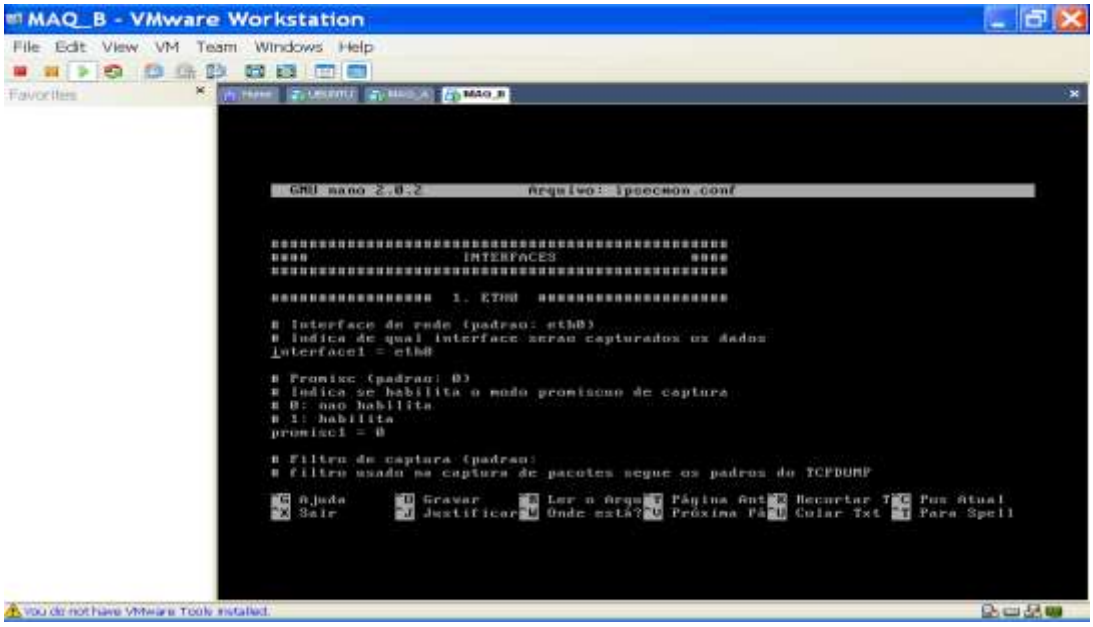

Figura 4.17 IPSecmon.conf Interfaces

Fonte: Autor 2008

Conforme figura 2.18 apresenta-se a inicialização do banco de dados MySQL com a tela phpMyAdmin 2.11.3deb1 ubuntu1.1

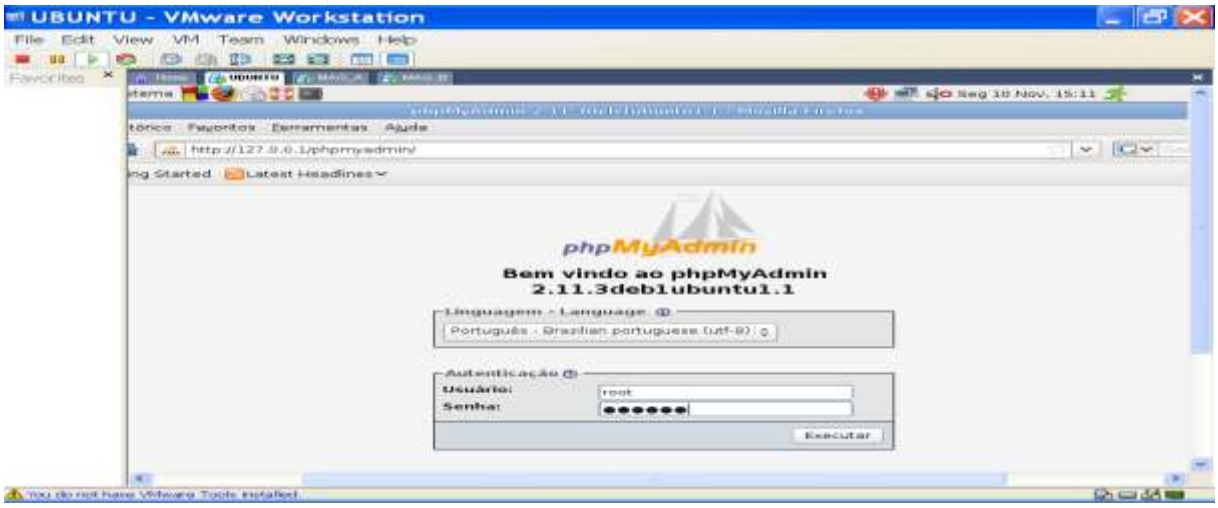

Figura 4.18 Banco de Dados tela de Login

Fonte: Autor 2008

Conforme figura 4.19 apresenta-se o banco de dados de nome agente, criado com suas respectivas tabelas de endereços e pacotes para ambas as máquinas A e B. Temos também o indicativo da tabela habilitado para acesso de inserção dos usuários.

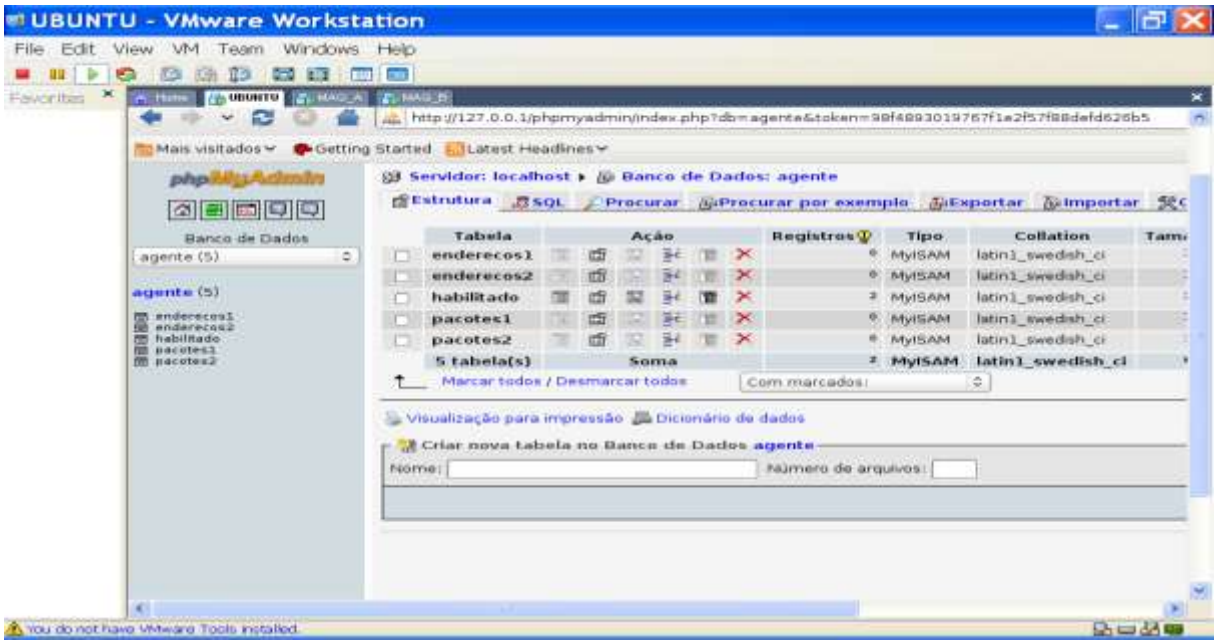

Figura 4.19 Banco de Dados Tabelas

Fonte: Autor 2008

Conforme figura 4.20 a tabela pacotes 1 e 2 apresenta-se os campos nos quais serão trabalhados na captura dos dados das suas respectivas máquinas:

a) ID – Identifica da entrada no banco

b) Cenário – Identifica qual arquivo será inserido

c) Date time – Dia e Hora

d) IPsrc – IP de origem

e) IPdest – IP de destino

f) Prot – Protocolo

g) Portsrc – Porta de origem

h) Portdst – Porta de destino

i) Tamanho – Tamanho do pacote

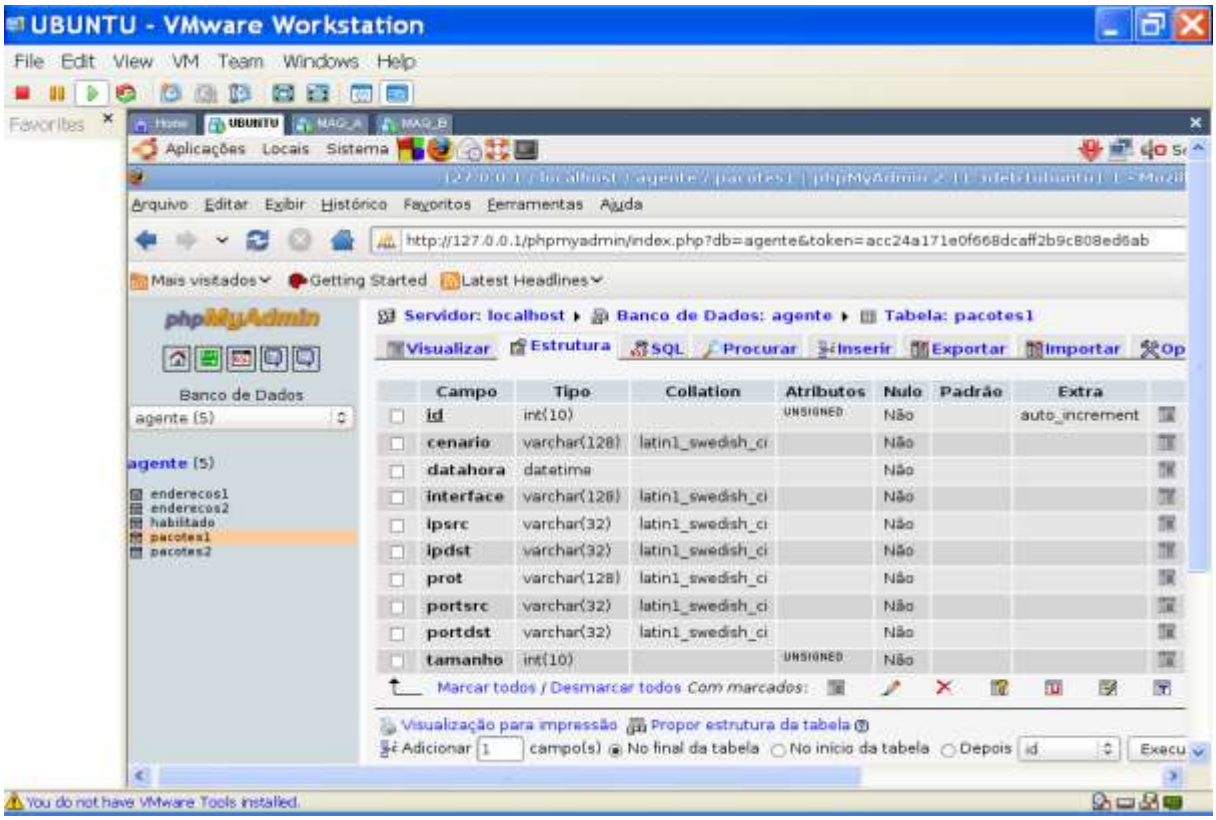

Figura 4.20 Banco de Dados Campos

Fonte: Autor 2008

Para a criação de gráficos foi utilizado o JPGraph 2.3.3 ferramenta escrita em PHP, no qual, oferece diversos recursos para manipulação de imagens. Com este recurso pode-se buscar dados no banco de dados e apresentar na tela em forma de gráfico conforme análise de desempenho capítulo 5.

Neste capitulo podemos observar as ferramentas que foram utilizadas para criação do túnel VPN e para criação da ferramenta de análise de desempenho contemplados na criação do túnel VPN e na criação IPSecMON. No capítulo 5 é observada a utilização destas ferramentas para análise do desempenho da informação trafegada em claro e cifrada.

5 ANÁLISE DE DESEMPENHO

Neste capítulo será abordada a análise de desempenho dos diversos cenários aplicados para identificação do consumo de tempo entre os pacotes trafegados em claro e cifrados. O desempenho será demonstrado graficamente para entendimento da relação de captura do banco de dados provenientes a inserção do agente inserido em cada máquina.

5.1 Demonstração da Análise de Desempenho

A) Etapa 1 – Zerando o banco de dados. Após zerar a captura do banco de dados no arquivo /etc/ipsecmon.conf conforme apresentado na figura 4.15, demonstra-se na figura 5.1 o gráfico inicial sem entrada de dados para que possa-se iniciar à tramitação, captura e inserção dos dados e popular o gráfico de forma demonstrativa de seu desempenho.

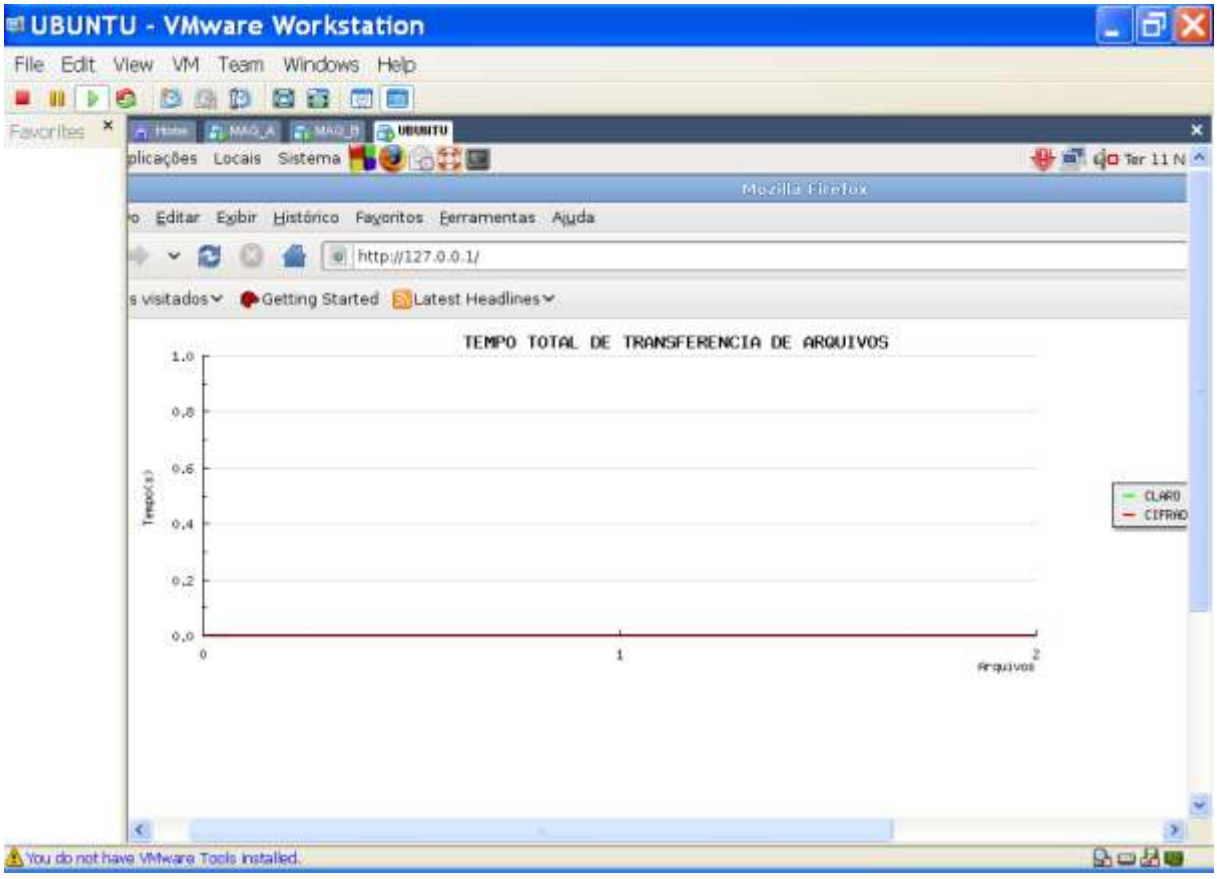

Figura 5.1 Gráfico Inicial

Fonte: Autor 2008

Após zerar o gráfico conforme etapa 1, inicializa-se o processo de tramitação dos dados em claro e cifrados com os cenários e interface devidamente atribuídos. Para que preencha o gráfico, deve-se dividir a tramitação em 2 conjuntos:

Conjunto 1 – Trafegar textos em claro. Deve-se desabilitar o túnel VPN com o comando /etc/init.d/ipsec stop, conforme figura 4.11. Trabalhar com os cenários 010MB, 050MB, 100MB, 200MB e 400MB dentro do arquivo /etc/ipsecmon.conf e com a interface 1 = eth0 atribuída no mesmo arquivo.

Conjunto 2 – Trafegar textos cifrados. Deve-se habilitar o túnel VPN com o comando /etc/init.d/ipsec start, conforme figura 4.7. Trabalhar com os cenários 010MB, 050MB, 100MB, 200MB e 400MB dentro do arquivo /etc/ipsecmon.conf e com a interface 1 = ipsec0 atribuída no mesmo arquivo.

A figura 5.2 apresenta-se a paralisação do IPSec, posteriormente o inicio de uma seção FTP para a tramitação dos arquivos de 10MB, 50MB, 100MB, 200MB e 400MB em claro. Usuário “teste” e senha “teste” foram utilizados para inicializar a sessão.

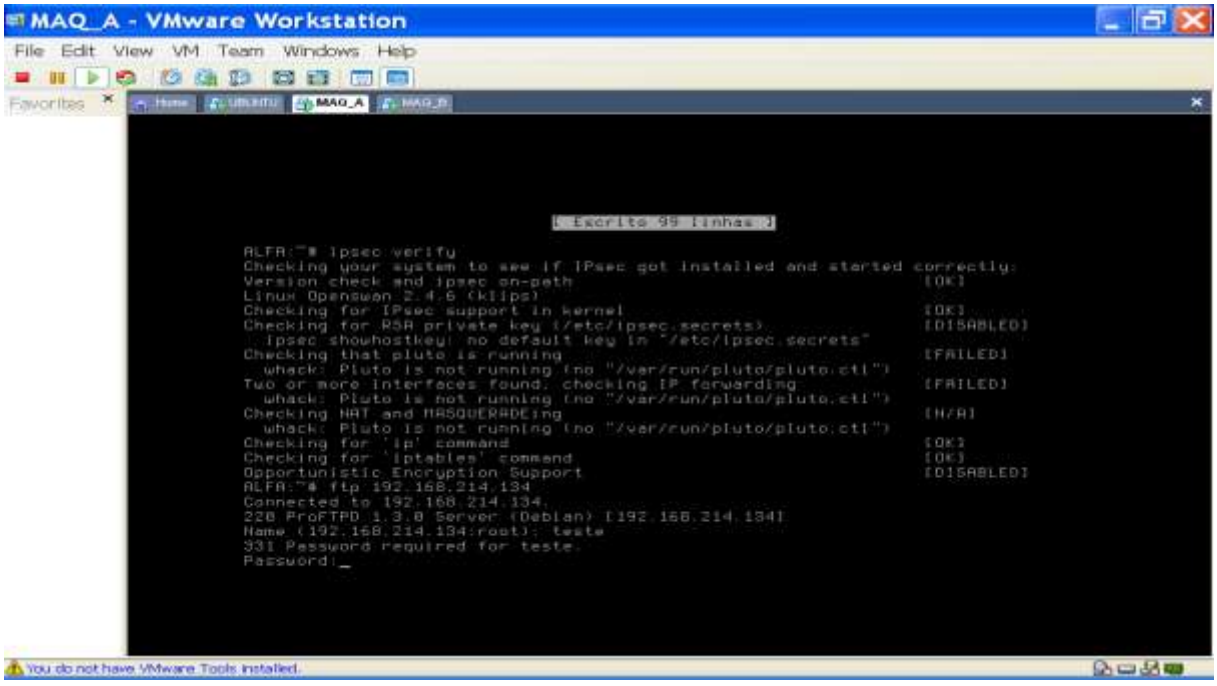

Figura 5.2 Inicialização do FTP

Fonte: Autor 2008

Etapa 2 – Transferência de arquivo de 10MB em claro. A figura 5.3 apresenta-se a tramitação do arquivo de cenário 010MB de 10MB para a primeira entrada no banco e atribuição na saída do gráfico.

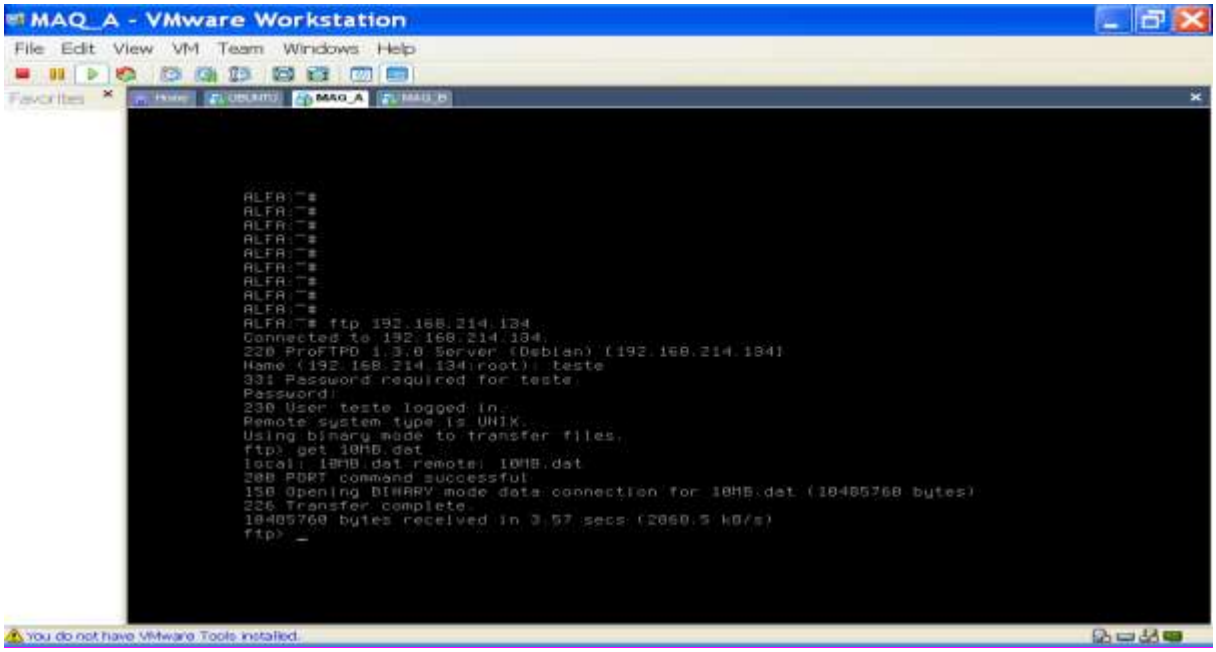

Figura 5.3 Transferência FTP 10MB

Fonte: Autor 2008

A figura 5.4 observa-se a entrada dos dados no banco de dados indicando o ID, cenário, data/hora, interface, ipsrc, ipdest, prot, portsrc, portdest e tamanho dos pacotes transitados.

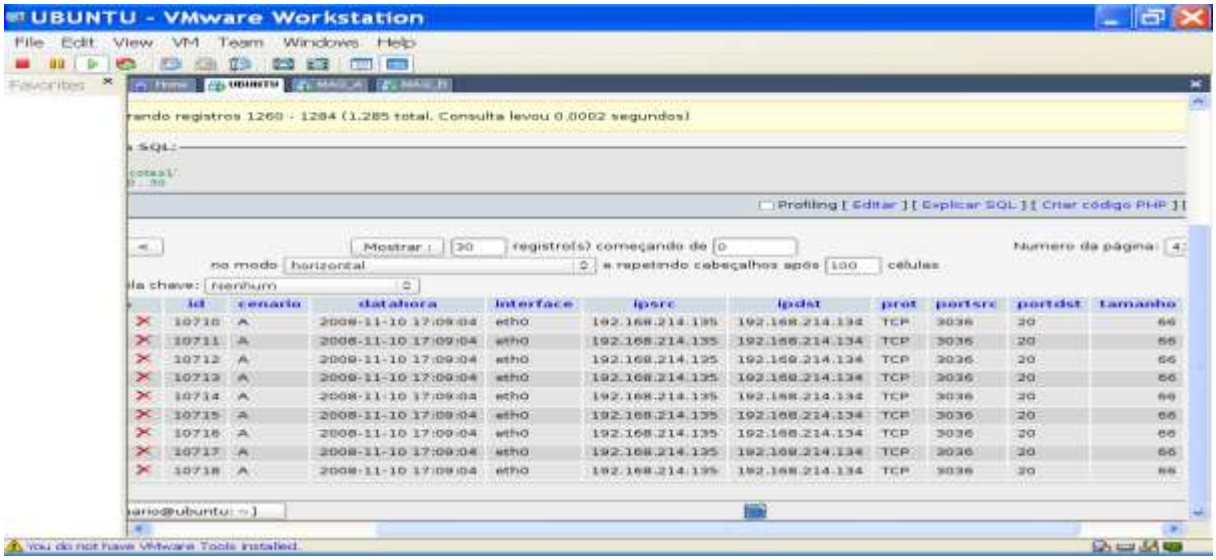

Figura 5.3 Transferência FTP 10MB

Fonte: Autor 2008

Conforme figura 5.5 observa-se que o tempo de transferência do arquivo de 10MB foi de 1 segundo para o trafego em claro conforme é demonstrado pela linha verde iniciada no eixo y. A indicação do arquivo está na posição 010 do eixo x.

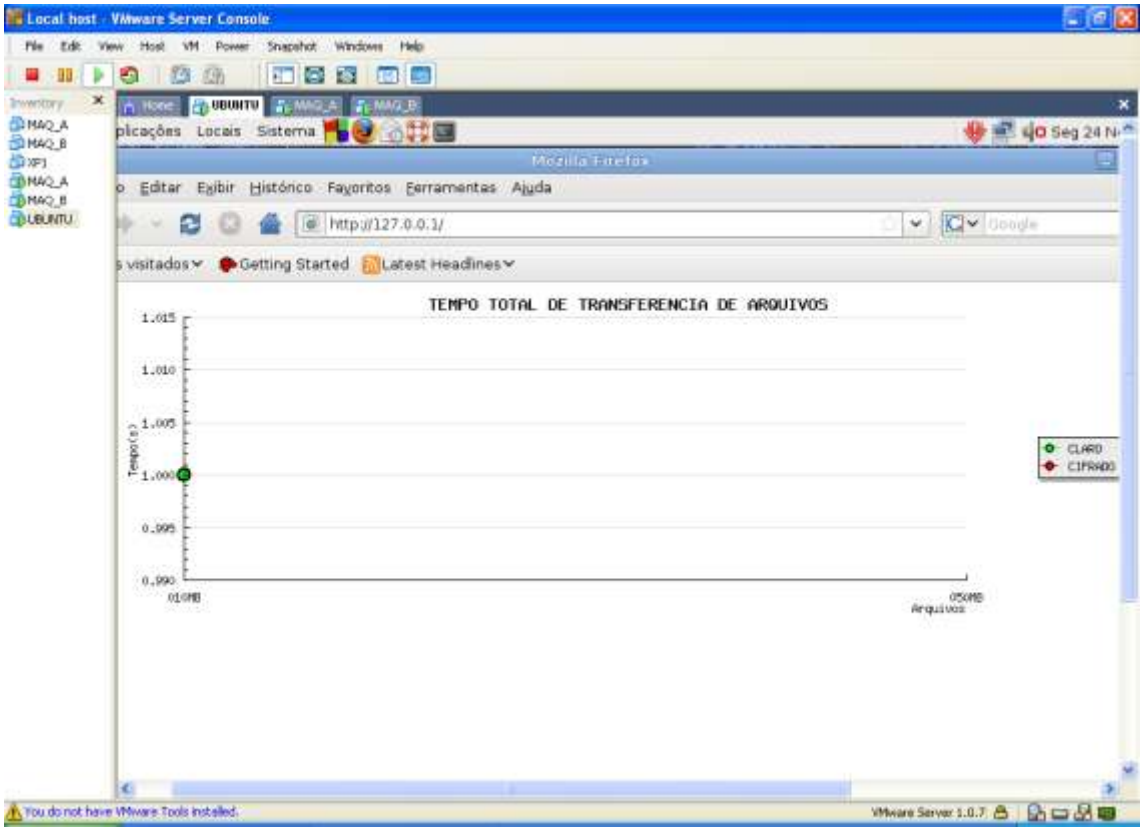

Figura 5.5 Gráfico Cenário 010MB claro

Fonte: Autor 2008

Etapa 3 – Transferência de arquivo de 50MB em claro. Observa-se no na figura 5.6 o tempo de transferência de 50MB do texto em claro de 5 segundos. O arquivo encontra-se na posição 050 do eixo x.

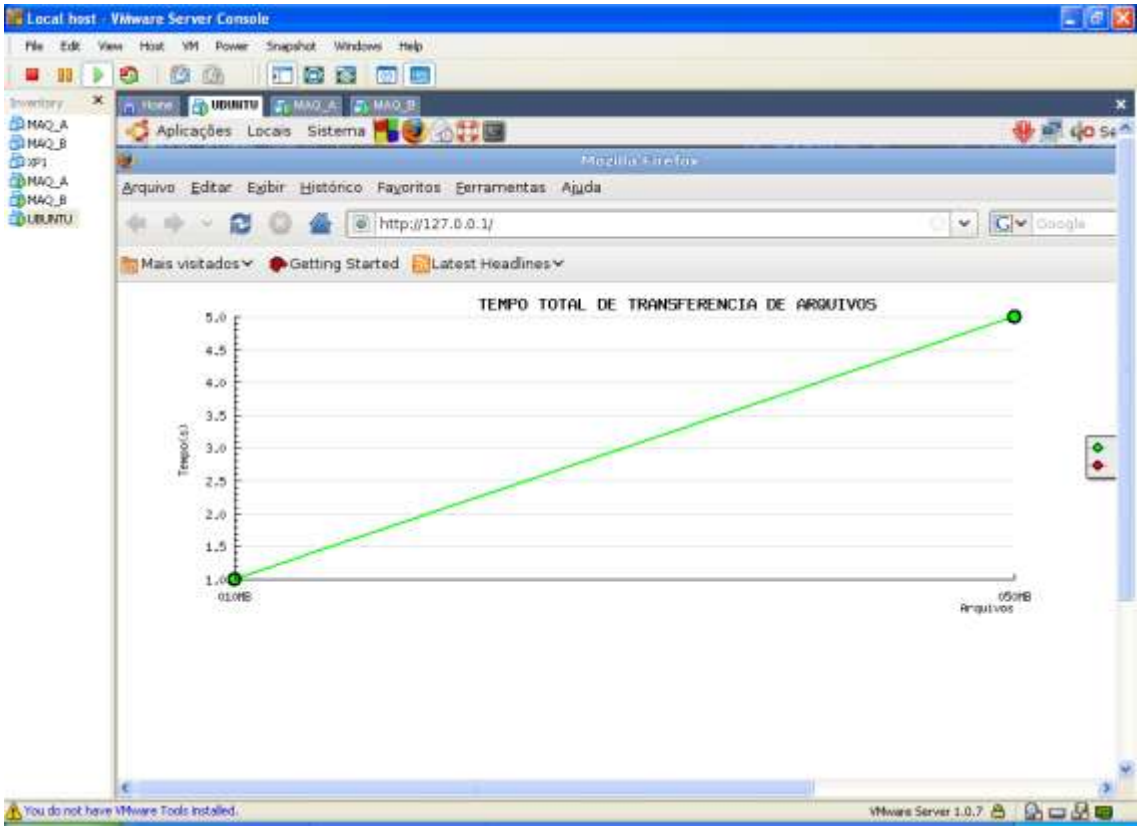

Figura 5.6 Gráfico Cenário 050MB

Fonte: Autor 2008

Etapa 4 – Transferência de arquivo de 100 MB em claro. Observa-se na figura 5.7 a transferência de 12 segundos do arquivo de texto em claro de 100 MB. O arquivo apresenta-se na posição 100MB do eixo x.

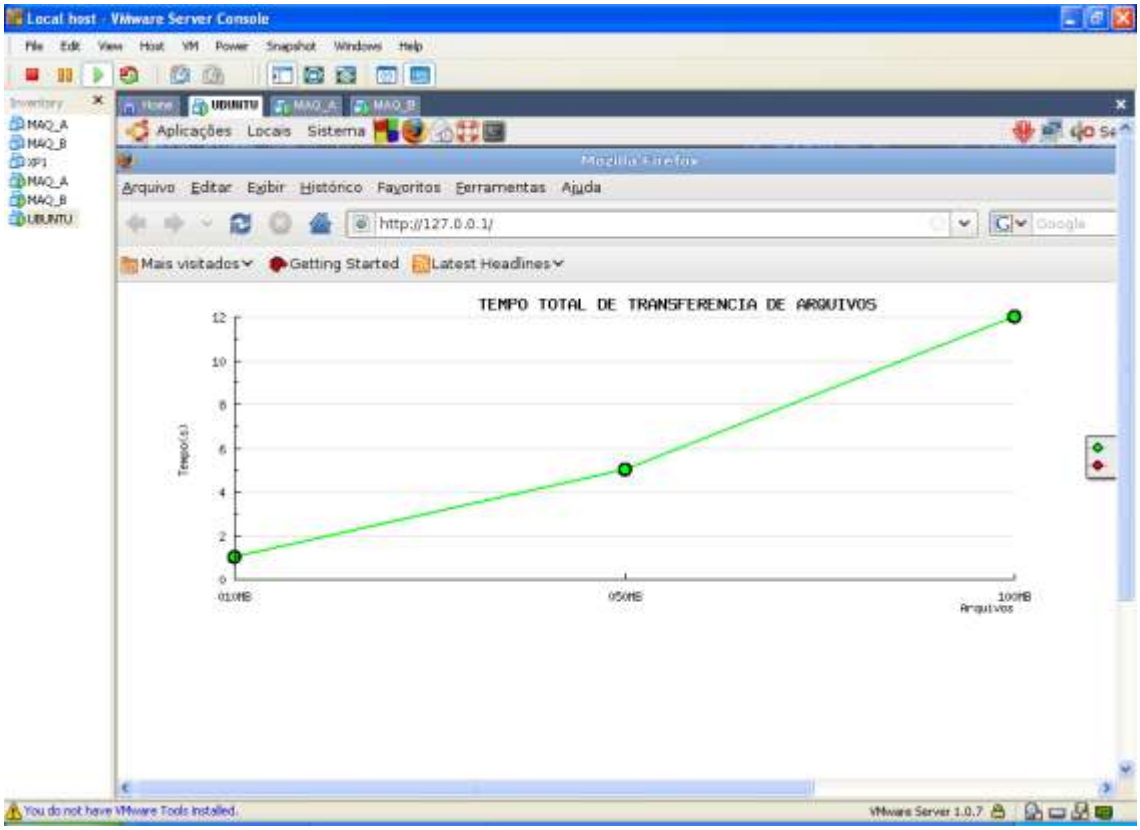

Figura 5.7 Gráfico Cenário 100MB claro

Fonte: Autor 2008

Etapa 5 – Transferência de arquivo de 200 MB em claro. Observa-se na figura 5.8 a transferência de 22 segundos do arquivo de texto em claro de 200 MB. O arquivo apresenta-se na posição 200MB do eixo x.

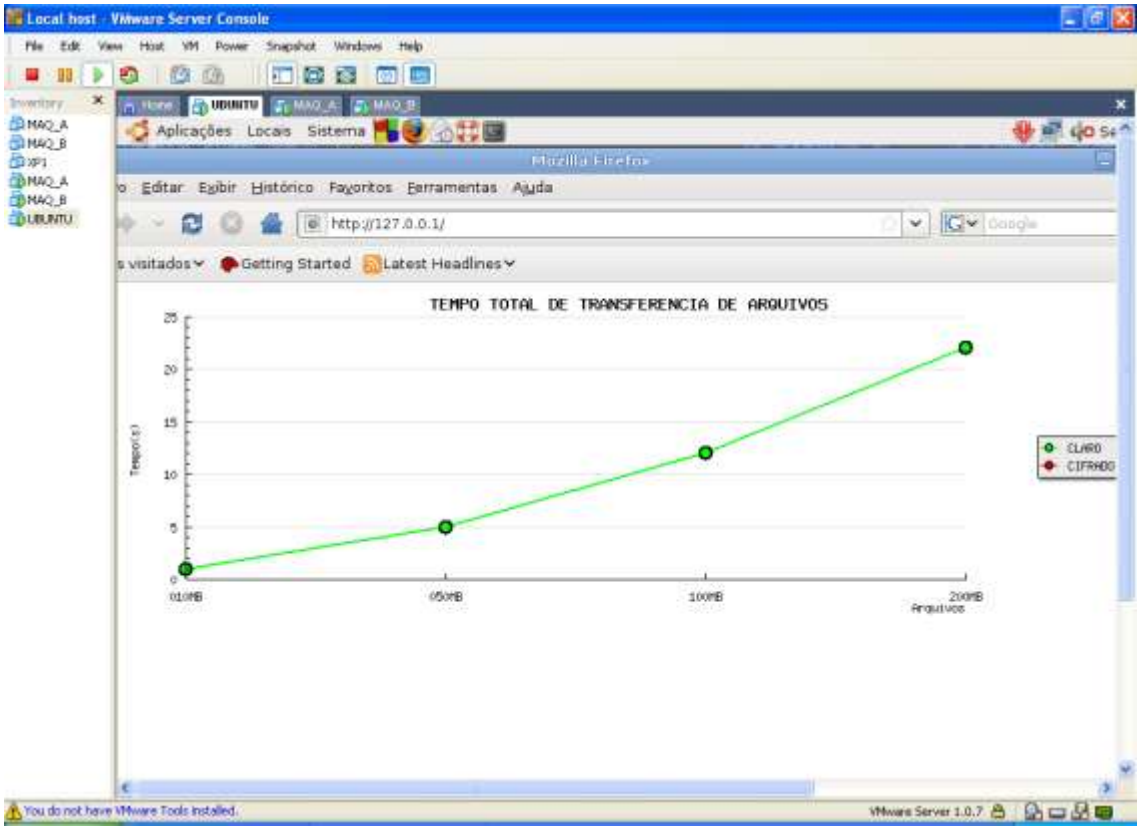

Figura 5.8 Gráfico Cenário 200MB claro

Fonte: Autor 2008

Etapa 6 – Transferência de arquivo de 400MB em claro. Observa-se na figura 5.9 a transferência de 41 segundos do arquivo de texto em claro de 400MB. O arquivo apresenta-se na posição 400MB do eixo x.

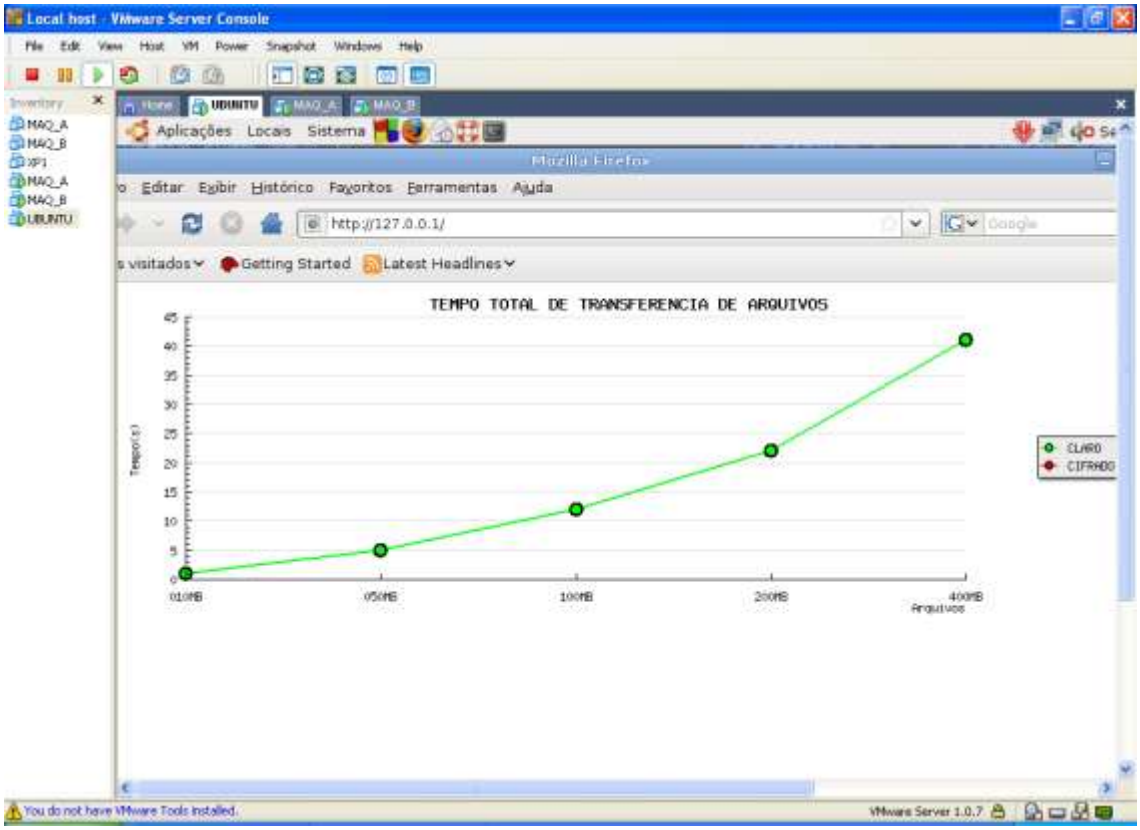

Figura 5.9 Gráfico Cenário 400MB claro

Fonte: Autor 2008

Etapa 7 – Transferência de arquivo de 10MB cifrado. Agora se observa a transferência do texto cifrado com o ponto vermelho no gráfico de 10MB, no inicio do eixo y, ocupando o tempo de 2 segundos na posição 010 do eixo x. Verifica-se inicialmente o dobro do tempo de transferência se comparar com o texto em claro de mesmo tamanho 10MB, conforme figura 5.10.

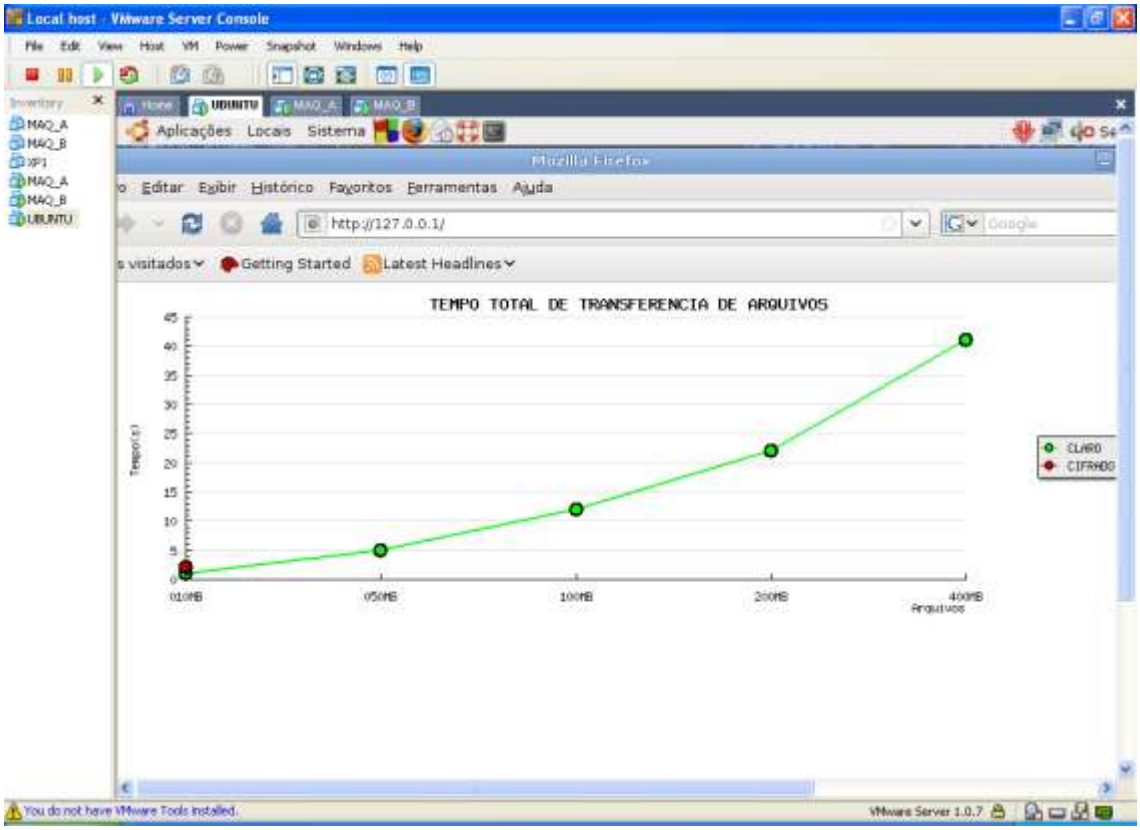

Figura 5.10 Gráfico Cenário 010MB cifrado

Fonte: Autor 2008

Etapa 8 – Transferência de arquivo de 50MB cifrado. No gráfico da figura 5.11 observa-se a transferência em 37 segundos de um arquivo de 50MB do texto cifrado no ponto 050MB do eixo x. A relação de transferência mais que quadriplicou em comparação de um arquivo de mesmo tamanho em texto em claro. Aparenta-se quanto maior o arquivo de mesmo tamanho cifrado maior o tempo de tramitação do mesmo.

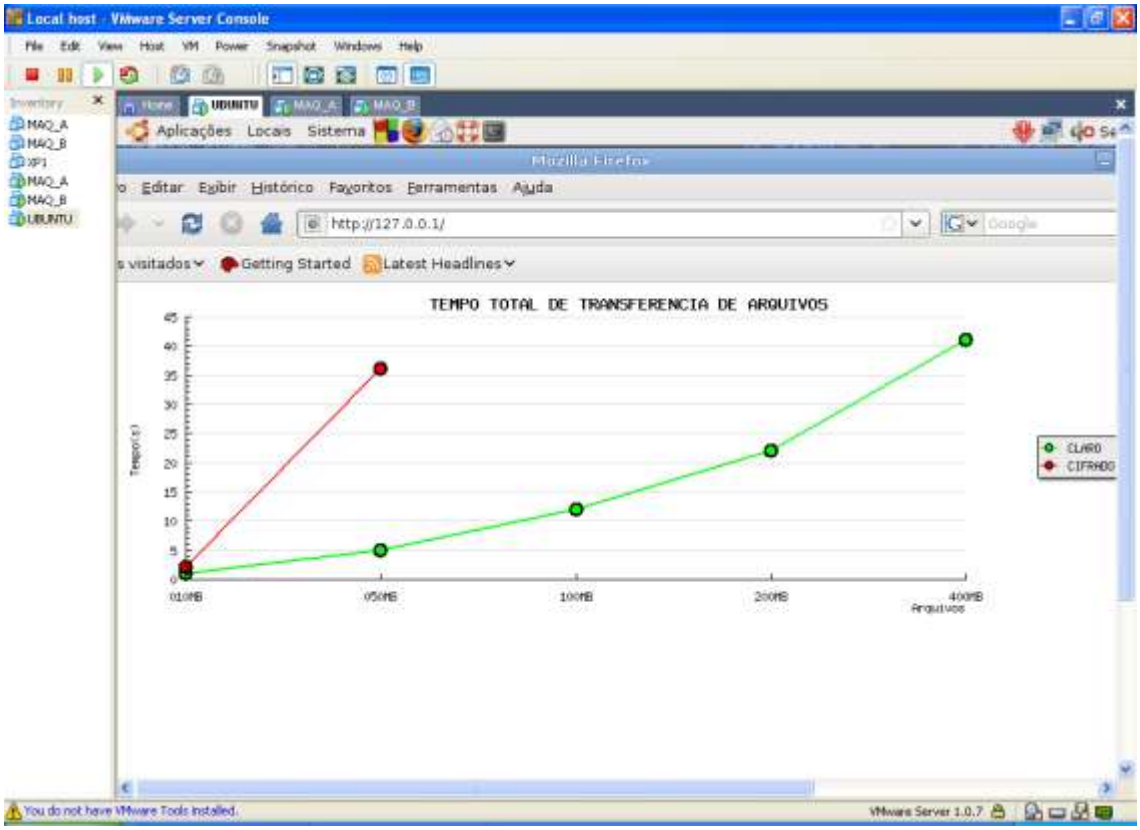

Figura 5.11 Gráfico Cenário 050MB cifrado

Fonte: Autor 2008

Etapa 9 – Transferência de arquivo de 100 MB cifrado. Na figura 5.12 observa-se o tempo de transferência em 117 segundos para um arquivo de 100 MB em texto cifrado, na posição 100MB do eixo x. Verifica-se que o tempo de transferência aumentou mais de 5 vezes em relação ao arquivo tramitado em claro.

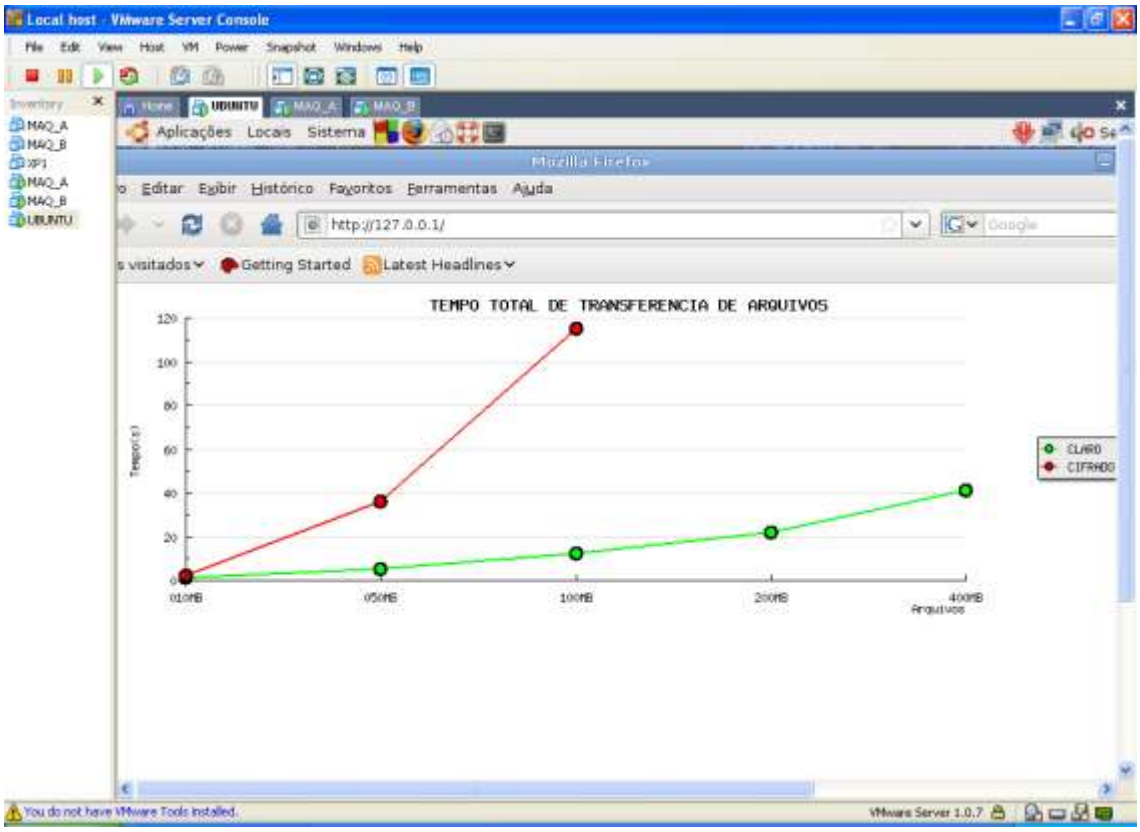

Figura 5.12 Gráfico Cenário 100MB cifrado

Fonte: Autor 2008

Etapa 10 – Transferência de arquivo de 200MB cifrado. Na figura 5.13 observa-se o tempo de transferência em 212 segundos para um arquivo de 200 MB em texto cifrado, na posição 200MB do eixo x. Verifica-se que o tempo de transferência aumentou mais de 6 vezes em relação ao arquivo de mesmo tamanho tramitado em claro.

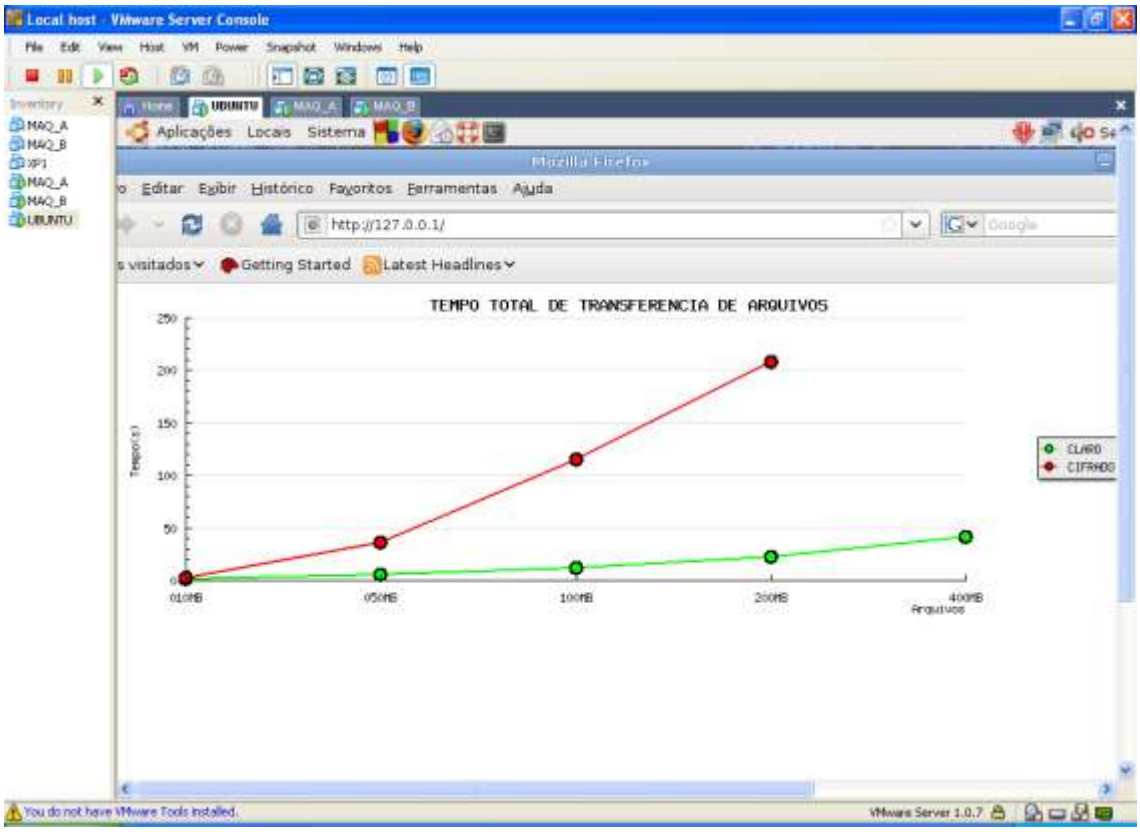

Figura 5.13 Gráfico Cenário 200MB cifrado

Fonte: Autor 2008

Etapa 11 – Transferência de arquivo de 400 MB cifrado. Na figura 5.14 observa-se o tempo de transferência em 390 segundos para um arquivo de 400 MB em texto cifrado, na posição 400MB do eixo x. Verifica-se que o tempo de transferência aumentou mais de 8 vezes em relação ao arquivo tramitado em claro.

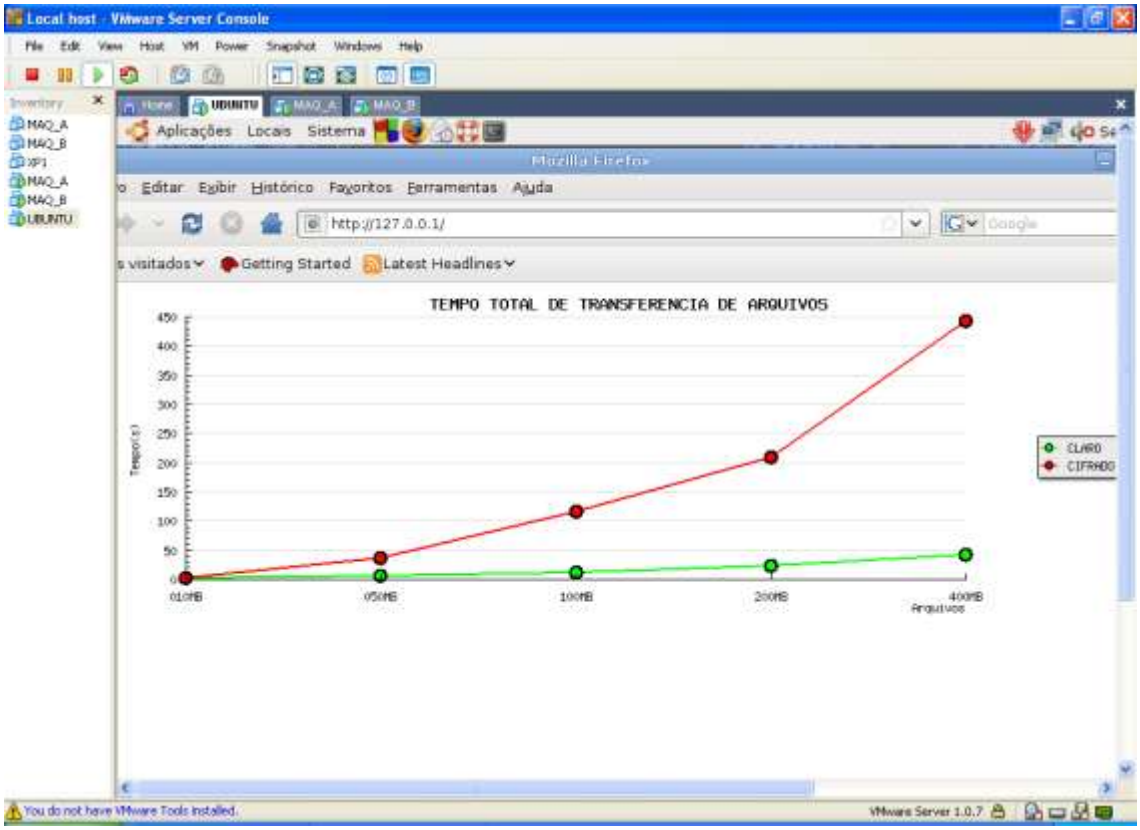

Figura 5.14 Gráfico Cenário 400MB cifrado

Fonte: Autor 2008

Tamanho do arquivo tramitado em MB	Tempo de transferência em segundos do dado em Claro	Tempo de transferência em segundos do dado Cifrado
10	1	2
50	5	37
100	12	117
200	22	212
400	41	390

Tabela 1 - Comparativo de Desempenho

Fonte: Autor 2008

Conforme a tabela 1, observa-se que quanto maior for o arquivo mais consumo de tempo ele terá e existe uma diferença enorme para a taxa de

transferência de um arquivo cifrado para um arquivo em claro isso se deve porque para os arquivos cifrados são adicionados mais pacotes e cabeçalhos a mais, no caso desse trabalho o cabeçalho esp. A diferença no arquivo de 10 Mb que é de apenas o dobro passa a ser o quádruplo no arquivo de 50 Mb e sei vezes mais quando o arquivo é de 100 Mb, ela é 9 vezes maior quando o arquivo é de 200 Mb e essa diferença tende a aumentar exponencialmente a medida que se aumenta o tamanho do arquivo.

Neste capítulo foi apresentada a análise de desempenho na tramitação dos textos em claro e dos textos cifrados. Para maior esclarecimento do conteúdo apresentado deve-se ter em mente o tempo de latência da rede, o tempo de processamento, memória e I/O da máquina por se tratar de todo processo de transferência ocorrer no mesmo computador em uma máquina virtual. Identificou-se uma comprovação de maior tempo de transferência para o arquivo cifrado em relação ao arquivo decifrado. Essa relação é acrescida com o tamanho de arquivo a ser tramitado.

6 CONCLUSÃO

O mundo tecnológico, cada vez mais utilizado nas necessidades cotidianas de transferência da informação entre diversas empresas e pessoas trás consigo as diversidades de potenciais perdas da informação em diversos âmbitos.

Neste trabalho podemos observar os aspectos de segurança, no qual, envolve a tramitação da informação entre pontos distintos. As vulnerabilidades apresentadas podem ser tratadas mediante a solução criptográfica demonstrada para o embaralhamento da informação tramitada. Provendo assim segurança da informação.

Apresentou-se também nessa monografia uma análise de desempenho em relação ao envio da informação trafegada em claro e cifrada. Está análise de desempenho demonstra um acréscimo de consumo de tempo, banda e recursos computacionais.

Os dados trafegados em claro minimizam a utilização dos recursos de máquina e banda de tráfego, mas por outro lado a informação trafegada está exposta aos riscos de serem identificadas, capturadas, manipuladas ou até mesmo roubadas. Já os dados trafegados de modo cifrados indicam um consumo maior do uso computacional e de utilização da banda, mas provendo a devida integridade, disponibilidade e confiabilidade para o armazenamento e a tramitação da informação.

Importa-se ter consciência de quanto vale a informação pessoal ou organizacional para que possamos mensurar qual a real necessidade da aplicabilidade dos métodos criptográficos. Assim tem-se a real identificação para aplicabilidade destes métodos apresentados para segurança da informação.

Para fazer esse trabalho precisei utilizar dos conhecimentos das disciplinas de banco de dados e redes.

REFERÊNCIAS BIBLIOGRÁFICAS

BEZERRA, Adonel. Club do Hacker. Disponível em: http://www.clubedohacker.com.br/tutoriaisartigos-mainmenu-31/26-pilha-tcpip/91-ataques-breve-descri- . Acesso 16 out. 2008.

CASE et al.,**RFC 3168**: The addition of explicit congestion notification (ECN) to IP status of this memo. sep./2001.

CASE et al. **RFC 791**: Internet protocol darpa internet program protocol specification. sep./1981.

CHESWICK, William R.; BELLOVIN, Steven M.; RUBIN, Aviel D. **Firewalls e segurança na internet** – repelindo o hacker ardiloso. 2 ed. Porto Alegre: Bookman, 2003.

COMER, Douglas E. **Interligação em Rede com TCP/IP** Volume 1 3 ed Rio de Janeiro: Campos, 1998

Internetworking With TCP/IP. 1.vol. 4. ed., New Jersey, EUA. Editora Prentice Hall, 2000

FERREIRA, F.N.F. **Segurança de informação**. Rio de Janeiro: C. Moderna, 2003.

GIL, Antônio Carlos. **Como Elaborar Projetos de Pesquisa**, 4. ed., São Paulo: Atlas, 2002

GOLDANI, Carlos Alberto. **IPSec e redes virtuais privadas** – informe técnico. Unicerte, 2004.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Fundamentos de Metodologia Científica,** 5ª. Ed., São Paulo, Atlas, 2003;

MORENO, Edward David; PEREIRA, Fábio Dacêncio; CHIARAMONTE, Rodolfo Barros. **Criptografia em Software e Hardware.** São Paulo: Novatec, 2005.

NAKAMURA, Emilio Tissato; GEUS, Paulo Lício de. **Segurança de Redes em Ambientes Cooperativos**. São Paulo: Novatec, 2007.

RAPPAPORT, Eduardo. Departamento de Engenharia Eletrônica e de Computação (DEL). **Escola politécnica**. UFRJ, 2003. Disponível em: http://www.gta.ufrj.br/~rezende/cursos/eel879/trabalhos/vpn/ipsec.html. Acesso 20 sep. 2008.

RHEE, Man Young. **Internet Security, Cryptografic Principles, Algorithms and Protocols.** Chichester, Inglaterra. Editora John Wiley and Sons, 2003

SILVA, Lino Sarlo da**. Public Key Infrastructure – PKI: Conheça a Infra-estrutura de Chaves Públicas e a Certificação Digital.** São Paulo: Novatec, 2004.

SILVA, Lino Sarlo da**. Virtual Private Network**. São Paulo: Novatec, 2003.

STALLINGS, William. **Criptografia e segurança de redes: Princípios e práticas**. 4 ed. São Paulo: Pearson Prentice Hall, 2008.

VOLPI, Marlon Marcelo. **Assinatura digital – Aspectos Técnicos, Práticos e Legais.** Rio de Janeiro: Axcel Books, 2001.

APÊNDICE A

Código da captura de pacotes, esse código desenvolvido em C tem como função a captura de pacotes e esses pacotes são jogados em um banco de dados são os agentes que ficam em cada máquina capturando e jogando para o banco de dados tudo o que passa pelo túnel VPN.

```
#include "pacotes.h"

/**
 * NOME:       pacotes_tcp
 * DESCRICAO:  Busca no pacote as portas de origem e destino TCP e as coloca em
 *             portsrc e portdst, respectivamente
 * RETORNA:    void (nada)
 * @param      packet  [u_char *]  conteudo do pacote total
 * @param      portsrc [char *]    recebera a porta origem
 * @param      portdst [char *]    recebera a porta destino
 * @param      len     [int]       indica o tamanho dos headers anteriores
 */
static void pacotes_tcp (const u_char * packet, char * portsrc,char * portdst, int len) {
   /// tcphdr:     estrutura de acesso ao conteudo TCP
   struct tcphdr *tcphdr = NULL;

   /// Faz a estrutura apontar para a posicao de inicio do TCP
   tcphdr = (struct tcphdr *)(packet+len);

   /// Armazena em portsrc e portdst os numeros das portas de origem e destino
   /// do pacote TCP, respectivamente
   sprintf(portsrc,"%d", ntohs(tcphdr->source));
   sprintf(portdst,"%d", ntohs(tcphdr->dest));

   /// retorna da funcao
   return;
}

/**
 * NOME:       pacotes_udp
 * DESCRICAO:  Busca no pacote as portas de origem e destino UDP e as coloca em
 *             portsrc e portdst, respectivamente
 * RETORNA:    void (nada)
 * @param      packet  [u_char *]  conteudo do pacote total
 * @param      portsrc [char *]    recebera a porta origem
 * @param      portdst [char *]    recebera a porta destino
 * @param      len     [int]       indica o tamanho dos headers anteriores
 */
static void pacotes_udp (const u_char * packet, char * portsrc,char * portdst, int len) {
```

```
    /// udphdr:     estrutura de acesso ao conteudo UDP
    struct udphdr *udphdr = NULL;

    /// Faz a estrutura apontar para a posicao de inicio do UDP
    udphdr = (struct udphdr *)(packet+len);

    /// Armazena em portsrc e portdst os numeros das portas de origem e destino
    /// do pacote UDP, respectivamente
    sprintf(portsrc,"%d", ntohs(udphdr->source));
    sprintf(portdst,"%d", ntohs(udphdr->dest));

    /// retorna da funcao
    return;
}

/**
  * NOME:       pacotes_ip
  * DESCRICAO:  Processa o cabecalho IP e o protocolo por ele encapsulado
  * RETORNA:    void (nada)
  * @param      arg     [u_char]        estrutura auxiliar de usuario
  * @param      pkthdr  [pcap_pkthdr *] ponteiro para o cabecalho do pacote
  * @param      packet  [u_char *]      conteudo do pacote total
  */
static void pacotes_ip ( u_char *arg,const struct pcap_pkthdr* pkthdr,const u_char * packet ) {
    /// buff:       buffer auxiliar, usado nas funcoes de log
    char buff[1024];

    /// estrutura auxiliar de usuario
    struct user *usuario;

    /// Faz a estrutura apontar para a posicao de inicio
    usuario= (struct user *) arg;

    /// contador:   contador de pacotes
    int contador =atoi(usuario->contador);

    /// iphdr:      estrutura de acesso ao conteuudo IP
    struct ip *iphdr = NULL;

    /// Faz a estrutura apontar para a posicao de inicio do IP
    iphdr = (struct ip *)(packet+14);

    /// ipdst:      recebera o IP destino
    char ipdst[50];

    /// ipsrc:      recebera o IP origem
    char ipsrc[50];

    /// ipprot:     recebera o codigo do protocolo encapsulado pelo IP
    char ipprot[50];

    /// ipprotnome: recebera o nome do protocolo encapsulado pelo IP
    char ipprotnome[50];

    /// iphl:       tamanho do header IP
    char iphl[50];

    /// portsrc:    recebera a porta de origem qdo o protocolo encapsulado for o
    ///             TCP ou UDP
    char portsrc[50]="0";
```

```
   /// portdst:    recebera a  porta  destino qdo o protocolo encapsulado for o
   ///             TCP ou UDP
   char portdst[50]="0";

   /// Armazena em ipdst o endereco ip destino
   sprintf(ipdst,"%s", inet_ntoa(iphdr->ip_dst));

   /// Armazena em ipsrc o endereco ip origem
   sprintf(ipsrc,"%s", inet_ntoa(iphdr->ip_src));

   /// Armazena em ipprot o codigo do protocolo encapsulado pelo IP
   sprintf(ipprot,"%02X", iphdr->ip_p);

   /// Armazena em iphl o tamanho do cabecalho IP
   sprintf(iphl,"%d", (iphdr->ip_hl+15));

   /// Armazena em ipprotnome o nome do protocolo encapsulado, e no caso de TCP
   /// ou UDP busca as portas de origem e destino
   switch (iphdr->ip_p) {
   case 0x01:
      sprintf(ipprotnome,"ICMP");
      break;
   case 0x02:
      sprintf(ipprotnome,"IGMP");
      break;
   case 0x06:
      sprintf(ipprotnome,"TCP");
      pacotes_tcp(packet,portsrc,portdst,(iphdr->ip_hl+15+14));
      break;
   case 0x08:
      sprintf(ipprotnome,"EGP");
      break;
   case 0x11:
      sprintf(ipprotnome,"UDP");
      pacotes_udp(packet,portsrc,portdst,(iphdr->ip_hl+15+14));
      break;
   case 0x32:
      sprintf(ipprotnome,"ESP");
      break;
   default:
      sprintf(ipprotnome,"IGNORADO");
   }

   /// Loga os dados obtidos do pacote
   sprintf(buff,"%d.      [%s]      %s:%s      >      %s:%s      (%s)",contador,usuario-
>interface,ipsrc,portsrc,ipdst,portdst,ipprotnome);
   log_entrada(buff);

   /// Armazena em banco de dados os dados obtidos do pacote
   if (atoi(bhab)==1) {
      /// SQL:    recebera o SQL e insercao no banco
      char SQL[1024];

      /// monta o SQL
      sprintf(SQL,"insert                    into                    pacotes%s
(cenario,datahora,interface,ipsrc,ipdst,prot,portsrc,portdst,tamanho)          values
('%s',current_timestamp,'%s','%s','%s','%s','%s','%s','%d')",bidt,cenario,usuario-
>interface,ipsrc,ipdst,ipprotnome,portsrc,portdst,pkthdr->len);
```

```
        /// Loga o SQL gerado
        /// sprintf(buff,"Banco: inserir (%s)",SQL);
        /// log_entrada(buff);

        /// executa a insercao em banco de dados
        int resultado=banco_inserir(SQL);

        if (resultado==1) log_entrada("Banco: ERRO NA INSERCAO");
    }

    /// retorna da funcao
    return;
}

/**
  * NOME:       pacotes_ethernet
  * DESCRICAO:  Processa o cabecalho ETHERNET e o protocolo por ele encapsulado
  * RETORNA:    void (nada)
  * @param      arg     [u_char]        estrutura auxiliar de usuario
  * @param      pkthdr  [pcap_pkthdr *] ponteiro para o cabecalho do pacote
  * @param      packet  [u_char *]      conteudo do pacote total
  */
static void pacotes_ethernet ( u_char *arg,const struct pcap_pkthdr* pkthdr,const u_char * packet) {
    /// buff:       buffer auxiliar, usado na funcao de log
    char buff[1024];

    /// estrutura auxiliar de usuario
    struct user *usuario;

    /// Faz a estrutura apontar para a posicao de inicio
    usuario= (struct user *) arg;

    /// contador:   contador de pacotes
    int contador =atoi(usuario->contador);

    /// etherhdr:   estrutura de acesso ao conteudo ETHERNET
    struct ether_header *etherhdr;

    /// Faz a estrutura apontar para a posicao de inicio do ETHERNET
    etherhdr= (struct ether_header *) packet;

    /// de acordo com o protocolo encapsulado pelo ETHERNET loga, e chama um
    /// processador no caso do IP
    switch (ntohs(etherhdr->ether_type)) {
    case 0x800:
        sprintf(buff,"%d. Ethernet IP Tamanho %d",contador,pkthdr->len);
        log_entrada(buff);
        pacotes_ip(arg,pkthdr,packet);
        break;
    case ETHERTYPE_ARP:
        sprintf(buff,"%d. Ethernet ARP Tamanho %d",contador,pkthdr->len);
        log_entrada(buff);
        break;
    default:
        sprintf(buff,"%d. Ethernet IGNORADO Tamanho %d",contador,pkthdr->len);
        log_entrada(buff);
    }

    /// retorna da funcao
    return;
```

```
}
/**
  * NOME:       pacotes_processar
  * DESCRICAO:  Funcao chamada pelo PCAP a cada pacote recebido, incrementa o
  *             contador de pacotes e reencaminha o pacote para o processador
  *             ETHERNET
  * RETORNA:    void (nada)
  * @param      arg     [u_char *]      estrutura auxiliar de usuario
  * @param      pkthdr  [pcap_pkthdr *] ponteiro para o cabecalho do pacote
  * @param      packet  [u_char *]      conteudo do pacote total
  */
static void pacotes_processar(u_char *arg,const struct pcap_pkthdr* pkthdr,const u_char * packet) {

    /// semaforo_up();
    /// estrutura auxiliar de usuario
    struct user *usuario;

    /// Faz a estrutura apontar para a posicao de inicio
    usuario= (struct user *) arg;

    /// contador:   contador de pacotes
    int contador = atoi(usuario->contador);

    /// incrementa o contador de pacotes
    contador++;

    /// armazena em arg[0] o contador
    sprintf(usuario->contador,"%d",contador);

    /// chama o processador de pacote ETHERNET
    pacotes_ethernet(arg,pkthdr,packet);

    /// semaforo_down();

    /// retorna da funcao
    return;
}

/**
  * NOME:       pacotes_parar
  * DESCRICAO:  Para a captura de pacotes
  * RETORNA:    void (nada)
  */
void pacotes_parar() {
    /// i:      variavel auxiliar de incremento
    int i;

    /// buff:   buffer auxiliar, usado nas funcoes de log
    char buff[1024];

    /// loga a chamada de parada de captura
    log_entrada("Pacotes: Loop parar");

    /// variavel global que controla o loop de captura
    LOOP=0;

    /// varre os descritores de interface e solicita a parada da captura
    for (i=0;i<MAX_INT;i++) {
        if (descritores[i]!=NULL) {
            /// loga a parada de captura
```

```
            sprintf(buff,"Pacotes: Parando o loop na interface (%s)",interfaces[i]);
            log_entrada(buff);
            /// para a captura
            pcap_breakloop(descritores[i]);
        }
    }

    /// Terminado o loop de captura, solicita o fechamento do banco de dados
    if (atoi(bhab)==1) {
        log_entrada("Banco: fechar");
        banco_fechar();
    }

    /// fecha os descritores das interfaces de captura
    pacotes_interfaces_fechar();

    /// liberar semaforo
    semaforo_fechar();

    /// loga o fim do loop de captura
    log_entrada("Pacotes: Fechando interfaces");

    /// Colocar marcador de fim no log
    log_fim();

    /// Finaliza o programa
    exit(0);

    /// retorna da funcao
    return;
}

/**
  * NOME:       pacotes_iniciar
  * DESCRICAO:  Inicia a captura de pacotes
  * RETORNA:    int
  */
int pacotes_iniciar() {
    /// buff:   buffer auxiliar, usado nas funcoes de log
    char buff[1024];
    /// errodispatch:   recebe o retorno da funcao pcap_dispatch (lpcap)
    int errodispatch=0;

    /// i:              variavel auxiliar de incremento
    int i=0;

    /// armazena em bhab se o banco de dados esta habilitado
    conf_banco_habilitado(bhab);

    /// armazena em bidt o identificador do usuario no banco de dados
    conf_banco_numero(bidt);

    /// armazena em cenario o identificador cenario avaliado
    conf_cenario(cenario);

    /// se banco de dados habilitado solicita a sua abertura (conexao)
    if (atoi(bhab)==1) {
        int resultado;

        log_entrada("Banco: conectar");
```

```
    banco_conectar();

    char SQL[1024];

    if (strcmp(cenario,"zerar")!=0) {

        /// monta o SQL
        sprintf(SQL,"delete from enderecos%s where cenario='%s'",bidt,cenario);
        /// Loga o SQL gerado
        sprintf(buff,"Banco: limpar enderecos (%s)",SQL);
        log_entrada(buff);
        /// executa a delecao em banco de dados
        resultado=banco_inserir(SQL);

        if (resultado==1) log_entrada("Banco: ERRO NA DELECAO da TABELA enderecos");


        /// monta o SQL
        sprintf(SQL,"delete from pacotes%s  where cenario='%s'",bidt,cenario);
        /// Loga o SQL gerado
        sprintf(buff,"Banco: limpar pacotes (%s)",SQL);
        log_entrada(buff);
        /// executa a delecao em banco de dados
        resultado=banco_inserir(SQL);

        if (resultado==1) log_entrada("Banco: ERRO NA DELECAO da TABELA pacotes");
    } else {
        /// monta o SQL
        sprintf(SQL,"delete from enderecos%s",bidt);
        /// Loga o SQL gerado
        sprintf(buff,"Banco: limpar todos os enderecos (%s)",SQL);
        log_entrada(buff);
        /// executa a delecao em banco de dados
        resultado=banco_inserir(SQL);

        if (resultado==1) log_entrada("Banco: ERRO NA DELECAO da TABELA enderecos");


        /// monta o SQL
        sprintf(SQL,"delete from pacotes%s",bidt);
        /// Loga o SQL gerado
        sprintf(buff,"Banco: limpar pacotes (%s)",SQL);
        log_entrada(buff);
        /// executa a delecao em banco de dados
        resultado=banco_inserir(SQL);

        if (resultado==1) log_entrada("Banco: ERRO NA DELECAO da TABELA pacotes");

        /// Fechar o banco de dados
        banco_fechar();

        /// finaliza o log
        log_fim();

        /// sai do programa
        exit(0);
    }

}
```

```
/// solicita a abertura das interfaces a serem capturadas, no caso de erro
/// fecha as interfaces ja abertas
if (pacotes_interfaces_abrir()==1) {
    pacotes_interfaces_fechar();
    return 1;
}

/// loga o inicio do loop de captura
log_entrada("Pacotes: Loop iniciado");

/// variavel global que controla o loop de captura
LOOP=1;

/// variavel que sera passada para os processador de pacotes
char dados_user[256];

/// estrutura auxiliar de usuario
struct user *usuario;

/// Faz a estrutura apontar para a posicao de inicio dos dados de usuario
usuario= (struct user *) dados_user;

/// armazena na posicao contador da estrutura de usuario 0
strcpy(usuario->contador,"0");

/// errbuf:     recebera o erro, se houver, da chamada de pcap_findalldevs
/// char errbuf[PCAP_ERRBUF_SIZE+1];

/// loop de captura
while (LOOP==1) {
    /// armazena na estrutura auxiliar de usuario o nome da interface
    strcpy(usuario->interface,interfaces[i]);

    /// chama o controlador da captura, a cada pacote recebido a funcao
    /// pacotes_processar eh chamada, passado o timeout determinado pela
    /// variavel refresh a funcao retorna
    errodispatch = pcap_dispatch(descritores[i],-1,pacotes_processar,(u_char *)dados_user);
    /// errodispatch = pcap_loop(descritores[i],-1,pacotes_processar,(u_char *)dados_user);

    /// int nonblock = pcap_getnonblock(descritores[i],errbuf);

    /// sprintf(buff,"Pacotes: lidos (%d) (%d)",errodispatch,nonblock);
    /// log_entrada(buff);

    if (errodispatch==-1){
        log_entrada("ERRO: erro na captura do pacote");
    }

    /// faz i indicar a proxima interface a ser capturada
    i=(i+1)%total_int;
}

/// Terminado o loop de captura, solicita o fechamento do banco de dados
if (atoi(bhab)==1) {
    log_entrada("Banco: fechar");
    banco_fechar();
}

/// fecha os descritores das interfaces de captura
pacotes_interfaces_fechar();
```

```
    /// loga o fim do loop de captura
    log_entrada("Pacotes: Loop finalizado");

    /// retorna da funcao
    return 0;
}

/**
  * NOME:       pacotes_interfaces_fechar
  * DESCRICAO:  Fecha os descritores das interfaces de captura de pacotes
  * RETORNA:    int
  */
int pacotes_interfaces_fechar() {
    /// i:  variavel auxiliar de incremento
    int i;

    /// buff:   buffer auxuliar, usado nas funcoes de log
    char buff[1024];

    /// varre os descritores de interfaces de captura e solicita o fechamento
    for (i=0;i<MAX_INT;i++) {
        if (descritores[i]!=NULL) {
            /// loga o fechamento
            sprintf(buff,"Pacotes: Fechando a interface (%s)",interfaces[i]);
            log_entrada(buff);
            /// fecha a interface de captura
            pcap_close(descritores[i]);
        }

    }

    /// retorna da funcao
    return 0;
}

/**
  * NOME:       pacotes_interface_dados
  * DESCRICAO:  Busca os IPs designados a uma interface
  * RETORNA:    int
  * @param  interface [char *]  nome da interface
  */
int pacotes_interface_dados(char * interface,char * ip) {
    /// alldevs:    recebera os detalhes de todas a interfaces em uma maquina
    pcap_if_t *alldevs;

    /// d:          ponteiro para uma das interfaces em alldevs
    pcap_if_t *d;

    /// a:          ponteiro para a estrutura de endereco de uma interface
    pcap_addr_t *a;

    /// addr:       estrutura auxiliar para obtencao do endereco IP
    struct in_addr addr;

    /// buff:       buffer auxiliar, usado nas funcoes de log
    char buff[1024];

    /// errbuf:     recebera o erro, se houver, da chamada de pcap_findalldevs
    char errbuf[PCAP_ERRBUF_SIZE+1];
```

```
    /// chama a funcao pcap_findalldevs (lpcap) para popular alldevs
    if (pcap_findalldevs(&alldevs, errbuf) == -1) {
        return 1;
    }

    /// varre as interfaces e obtem o endereco ip a elas designadas
    for (d=alldevs;d;d=d->next) {
        /// compara com o nome de interface recebido
        if (strcmp(interface,d->name)==0) {
            /// varre os enderco(s) IP
            for (a=d->addresses;a;a=a->next) {
                if (a->addr->sa_family==AF_INET)  {
                    /// obtem o endereco IP
                    addr.s_addr=((struct sockaddr_in *)a->addr)->sin_addr.s_addr;
                    sprintf(ip,"%s",inet_ntoa(addr));

                    /// loga o ip da interface
                    sprintf(buff,"Pacotes: Endereco IP da interface %s : %s",interface,ip);
                    log_entrada(buff);

                    break;
                }
            }
        }
    }
    /// retorna da funcao
    return 0;
}

/**
  * NOME:       pacotes_interfaces_abrir
  * DESCRICAO:  Abre as interfaces para captura
  * RETORNA:    int
  */
int pacotes_interfaces_abrir() {

    /// i:  variavel auxiliar de incremento
    int i=1;

    /// buff:   buffer auxiliar, usado nas funcoes de log
    char buff[1024];

    /// interface:  recebera o nome da interface
    char interface[100];

    /// filtro:     recebera o filtro a ser aplicado na captura
    char filtro[256];

    /// maxcapture:  recebera o tamanho maximo de captura
    char maxcapture[100];

    /// promisc:    recebera o valor que indica se a interface estara em modo
    ///             promiscuo
    char promisc[100];

    /// refresh:    tempo de captura e tempo de transporte de dados do kernel
    ///             para o nivel usuario
    char refresh[100];
```

```
    /// filtro_habilitado: recebera o valor que indica se o filtro sera aplicado
    char filtro_habilitado[100];

    /// errbuff:    buffer auxiliar, usado pcap_open_live caso ocorra um erro
    char errbuff[PCAP_ERRBUF_SIZE];

    /// datalink:  recebera o tipo da interface de rede
    int datalink;

    /// mascara de rede
    bpf_u_int32 netaddr=0, mask=0;

    /// filtro de pacotes
    struct bpf_program filter;

    /// total de interfaces habilitadas zerado
    total_int=0;

    /// varre o espaco de interfaces possiveis no arquivo de configuracao
    for (i=1;i<=MAX_INT;i++) {
       /// busca o nome da interface [i]
       conf_interface(i,interface);

       /// se encontrou
       if (strcmp(interface,"ERRO")!=0) {
          /// guarda no vetor de nomes de interfaces
          strncpy(interfaces[total_int],interface,100);
          /// busca filtro
          conf_filtro(i,filtro);
          /// busca habilita_filtro
          conf_filtro_habilitado(i,filtro_habilitado);
          /// busca maxcapture
          conf_maxcapture(i,maxcapture);
          /// busca habilita modo promiscuo
          conf_promisc(i,promisc);
          /// busca tempo de refresh
          conf_refresh(i,refresh);

          /// loga a tentativa de abertura da interface
          sprintf(buff,"Pacotes: Abrindo interface de captura (%s)",interface);
          log_entrada(buff);

          /// zera (null) o descritor na posicao total_int (contador de int ativas)
          descritores[total_int]=NULL;

          /// Abre interface para captura
          descritores[total_int] = pcap_open_live
(interface,atoi(maxcapture),atoi(promisc),atoi(refresh),errbuff);

          /// se deu erro na abertura loga
          if (descritores[total_int]==NULL) {
             sprintf(buff,"Erro: Nao foi possivel abrir a interface para captura (%s)",interface);
             log_entrada(buff);
             return 1;
          }

          /// loga teste de tipo de interface
          sprintf(buff,"Pacotes: Teste Ethernet (%s)",interface);
          log_entrada(buff);
```

```
        /// busca o tipo da interface, somente int. ethernet sao usadas
        datalink=pcap_datalink(descritores[total_int]);

        /// loga se deu erro
        if (datalink != DLT_EN10MB) {
           sprintf(buff,"Pacotes: Somente interfaces Ethernet sao usadas (%s)",interface);
           log_entrada(buff);
           return 1;
        }

        /// loga teste habilitacao de filtro
        sprintf(buff,"Pacotes: Verificando se filtro esta habilitado (%s)",interface);
        log_entrada(buff);

        /// se filtro habilitado
        if (atoi(filtro_habilitado)==1) {
           /// loga busca da mascara de rede
           sprintf(buff,"Pacotes: Buscando mascara de rede (%s)",interface);
           log_entrada(buff);

           /// busca a mascara de rede associada com uma interface
           pcap_lookupnet(interface,&netaddr,&mask,errbuff);

           /// Loga a compilacao do filtro
           sprintf(buff,"Pacotes: Compilando filtro (%s)",interface);
           log_entrada(buff);

           /// compila o filtro, se deu erro loga
           if (pcap_compile(descritores[total_int],&filter,filtro,1,mask)==-1) {
              sprintf(buff,"Erro: Nao foi possivel compilar o filtro de captura (%s
:%s)",interface,pcap_geterr(descritores[total_int]));
              log_entrada(buff);
              return 1;
           }

           /// loga a aplicacao do filtro na interface
           sprintf(buff,"Pacotes: Carregando filtro (%s)",interface);
           log_entrada(buff);

           /// aplica o filtro na interface, se deu erro loga
           if (pcap_setfilter(descritores[total_int],&filter)==-1) {
              sprintf(buff,"Erro: Nao foi possivel carregar o filtro de captura (%s:
%s)",interface,pcap_geterr(descritores[total_int]));
              log_entrada(buff);
              return 1;
           }
        } // teste filtro habilitado

        /// ip_int:     recebera o endereco ip relacionado com a interface
        char ip_int[32];
        /// busca o IP, se deu erro loga
        if (pacotes_interface_dados(interface,ip_int)==1) {
           sprintf(buff,"Erro: Nao foi possivel obter os dados da interface (%s)",interface);
           log_entrada(buff);
           return 1;
        }

        /// Armazena em banco de dados os dados obtidos do interface
        if (atoi(bhab)==1) {
           /// SQL:    recebera o SQL e insercao no banco
```

```
            char SQL[1024];

            /// monta o SQL
            sprintf(SQL,"insert into enderecos%s (cenario,datahora,interface,ip) values
('%s',current_timestamp,'%s','%s')",bidt,cenario,interface,ip_int);

            /// Loga o SQL gerado
            /// sprintf(buff,"Banco: inserir (%s)",SQL);
            /// log_entrada(buff);

            /// executa a insercao em banco de dados
            int resultado=banco_inserir(SQL);

            if (resultado==1) log_entrada("Banco: ERRO NA INSERCAO na TABELA enderecos");
         }

         pcap_setnonblock(descritores[total_int],1,errbuff);


         /// incrementa o total de interfaces ativas e capturando
         total_int++;
      }// teste interface encontrada em conf
   }// loop das interfaces

   /// retorna da funcao
   return 0;
}
```

APÊNDICE B

Código de demonstração em página, esse código desenvolvido em PHP tem como função pegar aquilo que esta no banco de dados que foi transferido pelo código de captura de pacotes e usar ele como saída para me gerar os gráficos em página.

```
<?php
include ("jpgraph/jpgraph.php");
include ("jpgraph/jpgraph_line.php");
include ("jpgraph/jpgraph_scatter.php");

include ("banco.php");
$conexao=conectar();

if (!$conexao) {
   echo "erro de bd";
   exit;
}
$min="";
$max="";
$max1=0;
$min1=0;
$max2=0;
$min2=0;
$tempo=0;

$contador=0;
```

```
$CLARO=array("");
$CIFRADO=array("");
$CENARIO=array("");

$sql="select cenario from pacotes1 union select cenario from pacotes2 order by cenario";
$consulta=executar($conexao,$sql);
while (buscar($consulta,$resultado)) {
   $contador++;
   $CENARIO[$contador]=$resultado[0];
}
liberar($consulta);

for ($i=1;$i<=$contador;$i++){
    /// CLARO
    $min1=-1;$max1=-1;
    $sql="select min(datahora),max(datahora) from pacotes1 where cenario='$CENARIO[$i]' and (portdst=20 or portsrc=20) and interface='eth0'";
    $consulta=executar($conexao,$sql);
    if (buscar($consulta,$resultado)) {
        $min = $resultado[0];
    $max = $resultado[1];
        $min1=strtotime($min);
    $max1=strtotime($max);
    }
    liberar($consulta);

    $min2=-1;$max2=-1;
    $sql="select min(datahora),max(datahora) from pacotes2 where cenario='$CENARIO[$i]' and (portdst=20 or portsrc=20) and interface='eth0'";
    $consulta=executar($conexao,$sql);
    if (buscar($consulta,$resultado)) {
        $min = $resultado[0];
    $max = $resultado[1];
        $min2=strtotime($min);
    $max2=strtotime($max);
    }
    liberar($consulta);

    if (($min1==NULL) || ($min2==NULL) || ($max1==NULL) || ($max2==NULL)) {
        $min=0;$max=0;
        $CLARO[$i]="";
    } else {
        $min=$min2;
        if ($min1<$min2) $min=$min1;

        $max=$max2;
        if ($max1>$max2) $max=$max1;

        $tempo=$max-$min;
        $CLARO[$i]=$tempo;
    }


    /// CIFRADO
    $min1=-1;$max1=-1;
    $sql="select min(datahora),max(datahora) from pacotes1 where cenario='$CENARIO[$i]' and (portdst=20 or portsrc=20) and interface='ipsec0'";
    $consulta=executar($conexao,$sql);
    if (buscar($consulta,$resultado)) {
```

```
        $min = $resultado[0];
    $max = $resultado[1];
        $min1=strtotime($min);
    $max1=strtotime($max);
    }
    liberar($consulta);

    $min2=-1;$max2=-1;
    $sql="select min(datahora),max(datahora) from pacotes2 where cenario='$CENARIO[$i]' and
(portdst=20 or portsrc=20) and interface='ipsec0'";
    $consulta=executar($conexao,$sql);
    if (buscar($consulta,$resultado)) {
        $min = $resultado[0];
    $max = $resultado[1];
        $min2=strtotime($min);
    $max2=strtotime($max);
    }
    liberar($consulta);

    if (($min1==NULL) or ($min2==NULL) or ($max1==NULL) or ($max2==NULL)) {
        $min=0;$max=0;
        $CIFRADO[$i]="";
    } else {
        $min=$min2;
        if ($min1<$min2) $min=$min1;

        $max=$max2;
        if ($max1>$max2) $max=$max1;
        $tempo=$max-$min;
        $CIFRADO[$i]=$tempo;
    }

}
fechar($conexao);
//echo "Claro:";
//print_r($CLARO);
//echo "Cifrado:";
//print_r($CIFRADO);

//exit;
//// GRAFICO

$LARGURA=900;
$ALTURA=300;

// Create the graph. These two calls are always required
$graph = new Graph($LARGURA,$ALTURA,"auto");
$graph->SetScale("textlin");
//$graph->yscale->SetAutoMin(0);

$graph->img->SetMargin(50,145,20,40);
$graph->SetFrame(false);
$graph->title->Set("TEMPO TOTAL DE TRANSFERENCIA DE ARQUIVOS");
$graph->xaxis->title->Set("Arquivos");
$graph->xaxis->SetTickLabels($CENARIO);
$graph->yaxis->title->Set("Tempo(s)");
$graph->yaxis->SetTitlemargin(40);


// Create the linear plot
```

```
$lineplot1=new LinePlot($CLARO);
$lineplot1->mark->SetType(MARK_FILLEDCIRCLE);
$lineplot1->mark->SetFillColor("green");
$lineplot1->mark->SetWidth(6);
$lineplot1->SetColor("green");
$lineplot1->SetWeight(2);
$lineplot1->SetLegend ("CLARO");

// Add the plot to the graph
$graph->Add($lineplot1);

// Create the linear plot
$lineplot2=new LinePlot($CIFRADO);
$lineplot2->mark->SetType(MARK_FILLEDCIRCLE);
$lineplot2->mark->SetFillColor("red");
$lineplot2->mark->SetWidth(6);
$lineplot2->SetColor("red");
$lineplot2->SetWeight(2);
$lineplot2->SetLegend ("CIFRADO");

// Add the plot to the graph
$graph->Add($lineplot2);


$graph->legend->Pos( 0.00,0.5,"right","center");
$handleIMG =  $graph->Stroke(_IMG_HANDLER);

$tmpfname = tempnam("tmp","gr_");
imagepng($handleIMG,$tmpfname);
$nome = basename($tmpfname);
$nome=base64_encode($nome);

echo "<img src='grafico.php?ID=".$nome."' style='margin-left: 50px;' width='$LARGURA'
height='$ALTURA' alt='Tempo de Transferencia' ><br><br>\n";



?>
```