

# EXTRAÇÃO DE ASSINATURAS DE PROTOCOLOS DE APLICAÇÃO ATRAVÉS DE ANÁLISE DE SUBSEQUÊNCIAS COMUNS

Grasielly Mayumi Sugimoto (Bolsista PIBIC/UEMS); Fabrício Sérgio de Paula  
Universidade Estadual de Mato Grosso do Sul  
Cidade Universitária de Dourados, Caixa postal 351, CEP: 79804-970  
[japa\\_grasy@hotmail.com](mailto:japa_grasy@hotmail.com); [fabricao@uems.br](mailto:fabricao@uems.br)

**Resumo:** *O nível de transporte contém a identificação das portas envolvidas na comunicação. Geralmente, as aplicações podem se comunicar em portas específicas de outras aplicações. Dessa forma, não se está totalmente seguro, pois o tráfego de uma determinada aplicação pode ser interpretado de uma forma ilegal. Este artigo apresenta uma forma de identificar protocolos de aplicação através do conteúdo do tráfego e não através das portas, analisando as subsequências mais comuns.*

**Palavras-chave:** *Tráfego. Portas específicas. Comunicação.*

**Abstract:** *The transport layer contains the identification of doors involved in communication. Generally, applications can communicate on specific ports of other applications. Thus, it is not entirely safe, because the traffic of a given application can be interpreted in an illegal fashion. This article presents a way-identified application protocols through the contents of the traffic, and not through the doors, analyzing the most common subsequences.*

**Key-words:** *Traffic. Specific ports. Communication.*

## 1. Introdução

A interconexão de computadores ou redes de computadores em uma instituição resulta em benefícios salientes, tais como a facilidade de comunicação, o compartilhamento de recursos, o aumento na confiabilidade computacional e a redução de custos. A Internet é, sem dúvida, o maior exemplo da interconexão de recursos computacionais, ultrapassando fronteiras geográficas, culturais e sociais. Entretanto, a diversidade verificada nesses ambientes interconectados contempla também indivíduos dispostos a promover perturbações.

Tais indivíduos são movidos por razões ideológicas, psicológicas, financeiras, desejo de vingança e até mesmo por pura diversão [Paula, 2004].

Por outro lado, assuntos relacionados à segurança computacional raramente recebem uma alta prioridade por parte de desenvolvedores de software, revendedores, administradores ou consumidores. Como resultado dessa distração, um número considerável de vulnerabilidades surge constantemente. Uma vez exploradas por atacantes, essas vulnerabilidades colocam instituições e usuários individuais sob risco [Spafford 1996, Pethia 2000].

Diversas tecnologias têm sido empregadas visando garantir os quesitos de segurança computacional [Nakamura/Geus, 2003]. *Firewalls* são aparatos muito utilizados há mais de uma década e consistem na análise de protocolos até o nível de transporte, decidindo que tipo de tráfego é permitido ou não em uma rede.

O nível de transporte — protocolos TCP ou UDP, na arquitetura TCP/IP — contém a identificação das portas envolvidas na comunicação. Dessa forma, os *firewalls* utilizam essa informação para bloquear certos tipos de aplicação indesejáveis em uma organização (*chat*, *ftp*, *p2p*, etc.).

Para contornar essa barreira, pelo menos dois mecanismos têm sido aplicados: 1) aplicações desenvolvidas para usar portas distintas das usuais; e 2) utilização de tunelamento dentro de outras aplicações. Dessa forma, um *firewall* comum não consegue identificar a aplicação real e acaba não funcionando de maneira adequada.

Uma solução para esse problema é ter um mecanismo que identifique a aplicação envolvida em uma comunicação através do conteúdo do tráfego na rede. Isso implica em encontrar assinaturas para protocolos de aplicação, que possam identificar as aplicações envolvidas sem o auxílio de números de portas. Nesse sentido, a identificação de subsequências de *bytes* comuns em tráfegos relacionados com um mesmo tipo de aplicação é um método que pode ser diretamente aplicado na extração de assinatura. As subsequências comuns encontradas são as assinaturas, que podem ser posteriormente usadas na identificação de protocolos.

## 2. Metodologia

O objetivo principal deste trabalho é estudar a aplicação do algoritmo LCSseq (*longest common subsequence*) na extração de assinaturas para protocolos de aplicação. Para tanto, é necessário conhecer sobre a especificação e funcionamento de alguns dentre os principais protocolos de aplicação utilizados. Após isso, é necessário estudar o funcionamento do

algoritmo LCSseq. Também deve-se verificar como a aplicação desse algoritmo implica na extração de boas assinaturas de protocolos. Essas assinaturas devem ser utilizados posteriormente para identificar as aplicações envolvidas em um determinado tráfego.

Para alcançar o objetivo principal, as seguintes atividades foram desenvolvidas:

1. Pesquisa sobre redes de computadores e arquitetura TCP/IP;
2. Estudo dos protocolos de aplicação FTP, TELNET e SSH;
3. Estudo dos protocolos de aplicação HTTP, SMPT e IMAP;
4. Estudo e implementação do algoritmo LCSseq;
5. Implementação de um protótipo para extração de assinaturas utilizando aplicações sucessivas do algoritmo LCSseq;
6. Realização de testes e coleta das assinaturas para os protocolos estudados;
7. Análise das assinaturas obtidas e elaboração de conclusões

### **3. Arquitetura de Redes**

Nos anos 60, o principal setor estratégico americano, *Department of Defense* – DoD se interessou em um protocolo que estava sendo desenvolvido/utilizado pelas universidades para interligação dos seus sistemas computacionais e que utilizava a tecnologia de chaveamento de pacotes. O problema maior estava na compatibilidade entre os sistemas computacionais de diferentes fabricantes que possuíam diferentes sistemas operacionais, topologias e protocolos. A integração e compartilhamento dos dados passaram a ser um problema de difícil resolução.

Foi atribuído assim à *Advanced Research Projects Agency* – ARPA a tarefa de encontrar uma solução para este problema de tratar com diferentes equipamentos e características computacionais. Foi feita então uma aliança entre universidades e fabricantes para o desenvolvimento de padrões de comunicação. Esta aliança especificou e construiu uma rede de teste de quatro nós, chamada ARPANET, e que acabou sendo a origem da Internet hoje. [Alves, 2000]

No final dos anos 70, esta rede inicial evoluiu, teve seu protocolo principal desenvolvido e transformado na base para o TCP/IP (*Transmission Control Protocol / Internet Protocol*). A aceitação mundial do conjunto de protocolos TCP/IP deveu-se principalmente a versão UNIX de Berkeley que além de incluir estes protocolos, colocava-os em uma situação de domínio público, onde qualquer organização, através de sua equipe técnica poderia modificá-los e assim garantir seu desenvolvimento.

O maior trunfo do TCP/IP é o fato destes protocolos apresentarem a interoperabilidade de comunicação entre todos os tipos de *hardware* e todos os tipos de sistemas operacionais.

Sendo assim, o impacto positivo da comunicação computacional aumenta com o número de tipos computadores que participam da grande rede Internet. [Alves, 2000]

A arquitetura TCP/IP, realiza a divisão de funções do sistema de comunicação em estruturas de camadas. Em TCP/IP as camadas são:

Aplicação
Transporte
Inter-Rede
Rede

- A camada de Rede é responsável pelo envio de datagramas construídos pela camada Inter-Rede. Esta camada realiza também o mapeamento entre um endereço de identificação de nível Inter-rede para um endereço físico ou lógico do nível de Rede. A camada Inter-Rede é independente do nível de Rede.
- A camada Inter-Rede realiza a comunicação entre máquinas vizinhas através do protocolo IP. Para identificar cada máquina e a própria rede onde estas estão situadas, é definido um identificador, chamado endereço IP, que é independente de outras formas de endereçamento que possam existir nos níveis inferiores. No caso de existir endereçamento nos níveis inferiores é realizado um mapeamento para possibilitar a conversão de um endereço IP em um endereço deste nível. Os protocolos existentes nesta camada são:
  - Protocolo de transporte de dados: IP - *Internet Protocol*;
  - Protocolo de controle e erro: ICMP - *Internet Control Message Protocol*;
  - Protocolo de controle de grupo de endereços: IGMP - *Internet Group Management Protocol*;
  - Protocolos de controle de informações de roteamento.
- A camada Transporte reúne os protocolos que realizam as funções de transporte de dados fim a fim, ou seja, considerando apenas a origem e o destino da comunicação, sem se preocupar com os elementos intermediários. A camada de transporte possui dois protocolos que são o UDP (*User Datagram Protocol*) e TCP (*Transmission Control Protocol*).
- A camada de Aplicação reúne os protocolos que fornecem serviços de comunicação ao sistema ou ao usuário. Pode-se separar os protocolos de aplicação em protocolos de serviços básicos ou protocolos de serviços para o usuário. Protocolos de serviços básicos, que fornecem serviços para atender as próprias necessidades do sistema de comunicação TCP/IP: DNS, BOOTP, DHCP. Protocolos de serviços para o usuário:

FTP, HTTP, TELNET, SMTP, POP3, IMAP, TFTP, NFS, NIS, LPR, LPD, ICQ, RealAudio, Gopher, Archie, Finger, SNMP e outros. [PUC]

### 3.1. Protocolos da camada Inter-Redes e Transporte

A camada inter-redes define um formato de pacote oficial e um protocolo chamado IP (*Internet Protocol*).

O protocolo IP realiza a função mais importante da camada Inter-Rede que é a própria comunicação inter-redes. Para isto ele realiza a função de roteamento, que consiste no transporte de mensagens entre redes e na decisão de qual rota uma mensagem deve seguir através da estrutura de rede para chegar ao destino.

Este protocolo utiliza a própria estrutura de rede dos níveis inferiores para entregar uma mensagem destinada a uma máquina que está situada na mesma rede que a máquina origem. Por outro lado, para enviar mensagem para máquinas situadas em redes distintas, ele utiliza a função de roteamento IP. Isto ocorre através do envio da mensagem para uma máquina que executa a função de roteador. Esta, por sua vez, repassa a mensagem para o destino ou a repassa para outros roteadores até chegar ao destino. [PUC]

Além da especificação formal e precisa de formatos de dados e de roteamento, o IP inclui um conjunto de regras que concentram a idéia da entrega não-confiável de pacotes. As regras definem como os *hosts* e os roteadores devem processar os pacotes, como e quando as mensagens de erro devem ser geradas e as condições segundo as quais os pacotes podem ser descartados. [Comer, 1998]

A camada de transporte define dois protocolos fim a fim, o primeiro deles é o TCP (*Transmission Control Protocol*) e o segundo é o UDP (*User Datagram Protocol*).

O protocolo TCP especifica o formato dos dados e das confirmações que os dois computadores trocam para oferecer uma transferência confiável e, também, os procedimentos de que se valem os computadores para assegurar que os dados cheguem corretamente.

Neste protocolo apresenta como principais características:

- Transferência de dados confiável fim a fim:
  - Todo pacote transmitido requer um *Ack* (é um bit de reconhecimento);
  - Há recuperação de dados perdidos;
  - Descarte de dados duplicados;
  - Reorganização dos dados recebidos fora de ordem.
- Comunicação bidirecional (*full-duplex*) entre cliente/servidor;

- O seqüenciamento: *bytes* de segmentos são numerados, de forma a garantir a entrega em ordem e a detecção e eliminação de duplicatas;
- É voltado para atuar sobre redes heterogêneas com tamanhos máximos de pacotes variáveis, faixas de passagem variáveis topologias distinto;
- O ponto fraco atual deste padrão é a adaptação a taxas de erros grandes, comum em comunicação sem fio (*wireless*).

Este protocolo divide o processo de comunicação em três fases: o início onde o cliente envia segmento tipo SYN (pedido de conexão, com número inicial da seqüência de numeração de *bytes* no sentido cliente/servidor), o servidor reconhece o pedido de conexão enviando segmento tipo SYN, com bit de reconhecimento (ACK) ligado e com número inicial de seqüência de numeração no sentido servidor/cliente, e o destino envia segmento ACK reconhecendo SYN do servidor, a troca de dados onde efetivamente ocorre a transferência de dados e o encerramento da conexão que pode ser iniciada tanto pelo cliente como pelo servidor e origem envia segmento FIN, o destino envia o reconhecimento: ACK e algum tempo depois a destino envia FIN (sinalizando fim da conexão) e por fim origem envia reconhecimento.

Quando cada segmento for transmitido é adicionado um *checksum* e quando estes são recebidos eles são verificados, se danificados os pacotes são descartados, como se tivessem se perdido pela rede e retransmitidos pela origem.

O protocolo TCP é fundamental para as comunicações da internet desde os primórdios da rede, pois a maior parte dos protocolos de aplicação necessita de transmissões confiáveis. Como consequência, a maioria dos protocolos de aplicação são implementados sobre TCP e não UDP e alguns poucos protocolos são implementados diretamente sobre IP. [Jannuzi, 2003]

O protocolo UDP fornece o mecanismo principal utilizado pelos programas aplicativos para enviar datagramas a outros programas iguais. O UDP fornece portas de protocolo para estabelecer a distinção entre os diversos programas executados em uma única máquina. Isto é, além dos dados enviados, cada mensagem UDP contém um número de porta de destino e também um número de porta de origem, tornando possível que o *software* UDP, no destino, entregue a mensagem ao destinatário certo e possibilitando, ao transportar uma mensagem de uma máquina à outra e, como o IP, fornece a mesma conotação de não-confiável de transmissão de datagrama sem conexão. Não usa confirmação para certificar-se de que as mensagens chegam, não ordena mensagens de entrada e não fornece informação para controlar a velocidade com que as informações fluem entre as máquinas. Assim, as

mensagens UDP podem se perder, ser duplicadas ou chegar com problemas. Mas ainda, os pacotes podem chegar mais rápidos do que podem ser processados pelo destinatário. [Comer, 1998]

### 3.2. Protocolos da camada de Aplicação

A camada de aplicação possui todos os protocolos de nível mais alto. Dentre eles estão o protocolo de transferência de arquivo FTP (*File Transfer Protocol*), o protocolo de terminal virtual TELNET, o protocolo de *login* remoto SSH, o protocolo usado para buscar páginas na *World Wide Web* HTTP (*HyperText Transfer Protocol*), o protocolo padrão para envio de *e-mails* SMTP (*Simple Mail Transfer Protocol*), o protocolo de gerenciamento de correio eletrônico IMAP (*Internet Message Access Protocol*), entre outros.

O protocolo FTP foi uma das primeiras aplicações na hoje chamada Internet. A base é o protocolo FTP que tem como principal função a transferência de arquivos entre dispositivos nos formatos ASCII e Binário. É uma aplicação do tipo cliente/servidor e em uma situação típica a aplicação cliente FTP utiliza o protocolo TCP para estabelecer uma conexão com o servidor remoto. Os servidores podem disponibilizar áreas só de leitura para *download* de arquivos compartilháveis ou leitura/escrita para áreas públicas sem restrição. [Alves, 2000]

Normalmente estes servidores permitem conexão autenticada, *login/senha*, com usuários cadastrados para acesso em áreas do servidor restritas ou ainda usuário anônimos ou mesmo FTP, com senha livre, normalmente o *e-mail*, para posterior contato. É importante observar que neste processo de autenticação o *login/senha* trafega pela rede sem criptografia, facilitando assim eventuais infortúnios como a utilização de analisadores de tráfego. Normalmente nos casos onde a autenticação é necessária se emprega servidores de FTP criptografados, sendo o *Security Shell* - SSH um dos mais populares.

Quando um cliente começa a negociar uma conexão com um servidor FTP, uma porta é escolhida e enviada para posterior conexão. O servidor, por sua vez, recebe a requisição pela porta padrão 20. A resposta do servidor é enviada pela porta 21 endereçada pela porta escolhida pelo cliente. A utilização do conceito de portas permite desta forma, que um mesmo servidor receba várias requisições, pois a resposta é endereçada à diferentes portas escolhidas por cada cliente. [Alves, 2000]

O protocolo TELNET é uma aplicação também do tipo cliente/servidor utiliza o protocolo TCP. É utilizada para conexão remota em computadores para execução de aplicações específicas muitas das vezes desenvolvidas pelo próprio usuário. [Alves, 2000]

Também usada para configuração e monitoramento remoto de equipamentos, como roteadores, por exemplo. Como não transfere arquivos, é comum a utilização de aplicações FTP ou TFTP em conjunto.

Da mesma forma que o FTP, existe a necessidade de autenticação e, portanto, todos os problemas relativos à segurança também estão presentes. Da mesma forma, existem aplicações TELNET criptografadas compartilhadas na Internet. [Alves, 2000]

O protocolo SSH é uma aplicação para *login* remoto seguro e outros serviços contextualizados em redes abertas. Esse protocolo é dividido em três componentes básicos:

- Camada de Transporte: Provê sigilo, integridade e autenticação do servidor. Opcionalmente, pode ser oferecido um serviço de compressão de dados. Essa camada geralmente é rodada em cima de uma conexão TCP/IP, mas é possível que ela seja usada em cima de qualquer outro tipo de protocolo confiável e orientado a conexão.
- Protocolo de Autenticação: Autentica o usuário perante o servidor. Esse protocolo necessita da camada de transporte para o seu funcionamento. [Sandes, 2003]
- Protocolo de Conexão: Permite a multiplexação da conexão entre vários canais lógicos. Esse protocolo depende do protocolo de autenticação. Nessa camada encontramos recursos como *shell* interativo, tunelamento de portas TCP/IP e conexões X11.

O SSH é bastante flexível no sentido de que os algoritmos usados em uma conexão são negociados entre o cliente e o servidor. Isso permite que algoritmos fracos sejam removidos ou novos algoritmos sejam testados sem que haja uma mudança no protocolo. [Sandes, 2003]

O protocolo do nível da aplicação para a transferência de hipertexto, o HTTP, opera sobre o protocolo TCP/IP para estabelecer um mecanismo de serviço com estrutura requisição-resposta. Uma das características peculiares de HTTP é a composição flexível do cabeçalho, composto por diversas linhas, o que permite sua utilização como integrador de diversos formatos e não apenas de documentos HTML.

Essa flexibilidade reflete-se também na maior complexidade desse protocolo. No entanto, é possível estabelecer servidores HTTP operando com configurações simplificadas, onde nem todos os serviços previstos no protocolo são implementados. [Ricarte, 2000]

Os principais serviços de HTTP incluem:

- GET: solicita ao servidor o envio de um recurso; é o serviço essencial para o protocolo;
- HEAD: variante de GET que solicita ao servidor o envio apenas de informações sobre o recurso;
- PUT: permite que o cliente autorizado armazene ou altere o conteúdo de um recurso mantido pelo servidor;
- POST: permite que o cliente envie mensagens e conteúdo de formulários para servidores que irão manipular a informação de maneira adequada;
- DELETE: permite que o cliente autorizado remova um recurso mantido pelo servidor.

O protocolo SMTP é a aplicação que permite transferir o correio de um servidor a outro em conexão não a ponto.

Trata-se de um protocolo que funciona em modo conectado, encapsulado numa trama TCP/IP. O correio é entregue diretamente ao servidor de correio do destinatário. O protocolo SMTP funciona graças aos comandos textuais enviados ao servidor SMTP. Cada um dos comandos enviados pelo cliente é seguido de uma resposta do servidor SMTP composta de um número e uma mensagem descritiva.

Eis um cenário de pedido de envio de *e-mail* a um servidor SMTP:

- Na abertura da sessão SMTP, o primeiro comando a enviar é o comando *HELO* seguida de um espaço (notado <SP>) e o nome de domínio da máquina (a fim de dizer “bom-dia, sou tal máquina”), seguidamente, validar por entrada (notado <CRLF>);
- O segundo comando é “*MAIL FROM*”, seguido do endereço do correio eletrônico do remetente. Se o comando for aceito, o servidor retorna a mensagem “250 OK”;
- O comando seguinte é “*RCPT TO:*”, seguido do endereço do correio eletrônico do destinatário. Se o comando for aceito o servidor retorna a mensagem “250 OK”;
- O comando *DATA* é a terceira etapa do envio. Anuncia o início do corpo da mensagem. Se o comando for aceito, o servidor retorna uma mensagem intermédia numerada 354 que indica que o envio do corpo do *e-mail* pode começar e considera o conjunto das linhas seguintes até ao fim da mensagem localizada por uma linha que contém unicamente um ponto. O corpo do *e-mail* contém eventualmente algumas das rubricas seguintes: *Date*, *Subject*, *Cc*, *Bcc*, *From*.

Se o comando for aceite o servidor retorna a mensagem "250 OK". [Comer, 1998]

O protocolo IMAP (*Internet Message Access Protocol*) trata-se de um método de acesso a mensagens eletrônicas armazenadas em um servidor local ou remoto.

Através de um programa cliente que envia comandos ao servidor de correio eletrônico que suporta o protocolo IMAP, o usuário pode manipular suas mensagens e pastas (também chamadas de *folders*) a partir de computadores diferentes em diversas localidades sem que seja necessário a transferências das mesmas do servidor para o computador de onde se está fazendo o acesso. Assim, as mensagens podem ser acessadas em um notebook durante uma viagem, no micro de casa ou do trabalho etc.

O protocolo IMAP prevê uma variedade de funcionalidades. Além das já oferecidas por protocolos como o POP3, destacam-se:

Acesso e manipulação de mensagens e de pastas de forma equivalente àquela feita em um acesso local;

- Possibilidade de acesso simultâneo a uma caixa postal compartilhada por mais de um usuário;
- Capacidade para que um programa cliente desconectado possa sincronizar seu conteúdo (mensagens, pastas e sub-pastas) com o do servidor;
- Ativar e desativar *flags* (marcações que indicam características de uma mensagem), que podem, inclusive, ser definidas pelo usuário. Com o POP3, estas marcações são registradas pelo cliente, de forma que, se a mensagem for aberta por um segundo cliente, as mesmas podem não ter seu status indicado corretamente. O IMAP permite a gravação das *flags* junto às caixas-postais, assegurando que, independente de qual cliente se acesse, as mensagens terão as mesmas corretamente atribuídas. [RNP, 1997]
- Capacidade de reconhecer os padrões de mensagens eletrônicas [RFC 822] e MIME-IMB [RFC 2045] em mensagens eletrônicas, de modo que os clientes de *e-mail* não o necessitem fazer. O servidor IMAP cumpre a tarefa de interpretar estes padrões, tornando os clientes mais fáceis de implementar e o acesso mais universal;
- Pesquisa de texto em mensagens de forma remota. Este modo de trabalho é feito localmente às caixas-postais;

A seleção para recebimento dos atributos de uma mensagem, ou seu texto ou anexos e outras partes (*attachments*) podem ser feitos de forma independente. Então, o usuário pode

pedir para receber de uma mensagem com um grande "attachment", apenas a parte do texto que lhe interessa o que é vantajoso no caso de um acesso discado de baixa qualidade. [RNP, 1997]

#### 4. Algoritmo LCSseq

##### 4.1. Algoritmo LCS

O algoritmo LCS (*Longest Common Subsequence*) extrai a maior subsequência comum entre duas seqüências.

A questão é geralmente definida como:

Dada duas seqüências, encontrar a maior subsequência presentes em ambos. Uma subsequência é uma seqüência que aparece na mesma ordem relativa, mas não necessariamente contínuas. Por exemplo, na seqüência ABCDEFG, "ABC", "ABG", "BDF", "AEG" são todos subsequências.

Quando o número de seqüências é constante, a questão é solúvel em tempo polinomial por programação dinâmica. Suponha que você tenha  $N$  seqüências, de comprimentos  $n_1, \dots, n_N$ . A pesquisa seria um teste para cada um dos  $2^{n_1}$  subsequências da primeira seqüência para determinar se eles são também subsequências do restante da seqüência; cada subsequência pode ser testada em tempo linear no comprimento do restante da seqüência, assim o tempo para este algoritmo seria:

$$O\left(2^{n_1} \sum_{i>1} n_i\right)$$

**Programação dinâmica** é uma técnica mais eficiente do que o método da força bruta, quando existe *overlap* de subproblemas, ótima infraestrutura e economiza espaço para melhorar o tempo complexidade de algoritmos.

O algoritmo LCS é um bom exemplo da programação dinâmica.

Dentre as várias aplicações, o algoritmo LCS é empregado na biologia computacional com a finalidade de encontrar uma semelhança entre duas cadeias de DNA. Com esse mesmo propósito de encontrar uma semelhança — padrão ou assinatura — é possível utilizá-lo para encontrar semelhanças entre duas porções de tráfego de rede.

Dada duas seqüências  $X$  e  $Y$ , uma seqüência  $Z$  é uma subsequência comum de  $X$  e  $Y$ , se  $Z$  é uma subsequência de  $X$  e de  $Y$  ao mesmo tempo.

Nesta questão temos duas seqüências e desejamos encontrar uma subsequência comum de comprimento máximo de  $X$  e de  $Y$  [Cormen, 2001].

Dada uma seqüência de símbolos  $X = x_1, x_2, \dots, x_m$ , uma subseqüência de  $X$  é uma seqüência que pode ser obtida a partir de  $X$  pela remoção de alguns símbolos. Mais precisamente, dada  $X = x_1, x_2, \dots, x_m$ , uma outra seqüência  $Z = z_1, z_2, \dots, z_k$  é uma subseqüência de  $X$  se existe uma seqüência de índices  $i_1, i_2, \dots, i_k$  estritamente crescente tal que para todo  $j = 1, 2, \dots, k$  temos que  $x_{i_j} = z_j$ .

Por exemplo, dado  $X = abcdbab$ , então  $Z = bcb$  é uma subseqüência de  $X$ . O Algoritmo 1 apresenta o procedimento LCS.

### **Algoritmo 1**

#### **Procedimento LCS (X, Y)**

**Entrada:** Duas cadeias de caracteres  $X$  e  $Y$  de tamanhos  $m$  e  $n$  respectivamente.

**Saída:** A matriz  $C$  indica o tamanho da maior subseqüência comum e a matriz  $B$  indica a solução ótima da LCS.

**início**

**para**  $i$  **de** 1 **até**  $m$  **faça**

$C[i, 0] \leftarrow 0;$

**para**  $j$  **de** 0 **até**  $n$  **faça**

$C[0, j] \leftarrow 0;$

**para**  $i$  **de** 1 **até**  $m$  **faça**

**para**  $j$  **de** 1 **até**  $n$  **faça**

**se**  $x_i = y_j$  **então**

$C[i, j] \leftarrow C[i - 1, j - 1] + 1;$

$B[i, j] \leftarrow \text{“}\nwarrow\text{”};$

**senão**

**se**  $C[i - 1, j] \geq C[i, j - 1]$  **então**

$C[i, j] \leftarrow C[i - 1, j];$

$B[i, j] \leftarrow \text{“}\uparrow\text{”};$

**senão**

$C[i, j] \leftarrow C[i, j - 1];$

$B[i, j] \leftarrow \text{“}\leftarrow\text{”};$

**retorne**  $C, B;$

**fim.**

Ao utilizar a técnica de programação dinâmica para resolver esta questão obtém-se tempo de execução para encontrar o comprimento de um LCS é  $O(mn)$ , onde  $m$  é o comprimento da cadeia  $X$  e  $n$  é o comprimento da cadeia  $Y$  [Cormen, 2001].

**Exemplo:**

Considere:

$X = ABCB$

$Y = BDCAB$

		j					
		0	1	2	3	4	5
		Y <sub>j</sub>	B	D	C	A	B
i	X <sub>i</sub>	0	0	0	0	0	0
1	A	0	0	0	0	1	1
2	B	0	1	1	1	1	2
3	C	0	1	1	2	2	2
4	B	0	1	1	2	2	3

LCS (Ordem inversa): B C B

**4.2. LCSseq**

Neste trabalho, o interesse é encontrar assinaturas utilizando diversas porções de tráfego de rede e não somente duas porções, como o LCS faz. Para tanto, é necessário utilizar um algoritmo para a questão nLCS, que é encontrar a maior subsequência comum entre  $n$  cadeias de caracteres. Porém, essa solução é NP-Difícil [Maier 1978], o que torna seu uso inviável. Existe um algoritmo probabilístico para a questão nLCS, que foi inicialmente estudado, mas não implementado e testado durante este trabalho [Bonizzoni et al. 2001].

Tendo em vista a dificuldade da questão nLCS, optou-se por utilizar a idéia de extrair subsequências comuns apenas como um parâmetro de comparação. Dessa forma, foi proposto e utilizado o procedimento nCS (*n Common Subsequence*), conforme o Algoritmo 2.

**Algoritmo 2**

**Procedimento** nCS ( $A_1, A_2, \dots, A_n$ )

**Entrada:**  $A_1, A_2, \dots, A_n$ , que são as  $n$  cadeias de caracteres.

**Saída:** C que representa uma subsequência comum entre as n cadeias de entrada.

**início**

$C \leftarrow \text{LCS}(A_1, A_2);$

**para**  $i \leftarrow 3$  **até**  $n$  **faça**  $C \leftarrow \text{LCS}(C, A_i);$

**fim.**

Note que nCS utiliza o algoritmo LCS iterativamente. Ele extrai uma subsequência comum iterativamente, utilizando o LCS para extrair a maior subsequência comum entre a cadeia atual e a próxima cadeia. O resultado é possivelmente uma cadeia vazia.

Esse algoritmo é eficiente, mas não garante a extração da maior subsequência comum entre as n cadeias de caracteres. Em outras palavras, ele não resolve a questão nLCS. Entretanto, ele é utilizado como um parâmetro para ser comparado com outro algoritmo. Embora não resolva a questão nLCS, o algoritmo nCS trouxe alguns resultados surpreendentes, considerando a sua simplicidade.

O algoritmo nCS é sinônimo do algoritmo LCSseq.

## 5. Resultados práticos

Nesta seção, são apresentados os resultados obtidos com os testes feitos com os algoritmos LCS e nCS. A implementação foi realizado em linguagem C++ utilizando o compilador GNU/g++.

Foi utilizada a ferramenta *tcpflow* para ler os arquivos no formato do *tcpdump* e gerar vários arquivos contendo os dados das conexões existentes, neste projeto, o conjunto de dados coletados, apresenta pacotes de 4 aplicações: FTP, HTTP, SMTP e TELNET. O formato dos nomes dos arquivos gerados pelo *tcpflow* é: ip\_de\_origem: porta\_de\_origem - ip\_de\_destino: porta\_de\_destino.

Foi utilizado também o comando *time* do Linux com a finalidade de medir o tempo de execução durante a extração das assinaturas e a detecção dos protocolos de aplicação.

Na tabela 1 são apresentadas as assinaturas obtidas ao aplicar o algoritmo nCS, considerando os primeiros 100 bytes de cada conexão. O tráfego utilizado aqui foi retirado da primeira semana de captura dos *datasets* do DARPA.

**Tabela 1. Extração de assinatura, na primeira semana, utilizando o algoritmo nCS**

<b>Aplicações</b>	<b>Conexões</b>	<b>Assinatura extraída</b>	<b>Tempo</b>
FTP	800	<i>nm</i>	<i>0m1,75 s</i>
HTTP	25.965	<i>T/</i>	<i>3m46,50 s</i>
SMTP	4.614	<i>(vazia)</i>	<i>0m22,48 s</i>
TELNET	1.974	.....	<i>0m4,32 s</i>

A tabela está organizada da seguinte forma: na primeira coluna identifica qual é a aplicação envolvida; na segunda coluna, a quantidade de conexões utilizadas para a extração de assinatura; na terceira coluna é apresentada as assinaturas extraídas no formato ASCII; a última coluna apresenta o tempo decorrido durante o processo de extração.

A próxima etapa foi apresentar as assinaturas obtidas usando o tráfego retirado da terceira semana de captura dos *datasets* do DARPA, diferente dos utilizados para extração da primeira assinatura, como mostra a tabela 2.

**Tabela 2. Extração de assinatura, na terceira semana, utilizando o algoritmo nCS**

<b>Aplicações</b>	<b>Conexões</b>	<b>Assinatura extraída</b>	<b>Tempo</b>
FTP	534	<i>S</i>	<i>0m2,60 s</i>
HTTP	26.712	<i>T</i>	<i>2m37,92 s</i>
SMTP	3.974	<i>R</i>	<i>0m16,12 s</i>
TELNET	828	.....	<i>0m6,79 s</i>

De acordo com as tabelas 1 e 2, mostra que o nCS não obteve uma assinatura padrão, pois, como já foi citado no item anterior, o nCS extrai uma subsequência comum iterativamente, utilizando o LCS para extrair a maior subsequência comum entre a cadeia atual e a próxima cadeia. O resultado seria possivelmente uma cadeia pequena ou vazia, o que comprova as tabelas.

## **6. Discussão**

Com base no objetivo deste trabalho, foi realizado o estudo teórico das atividades relacionadas, como: pesquisa sobre a arquitetura TCP/IP, estudo dos protocolos FTP, TELNET, SSH, HTTP, SMTP, IMAP e o estudo e implementação do algoritmo LCSseq, implementação de um protótipo para extração de assinatura utilizando aplicações sucessivas

do algoritmo LCSseq, testes e coletas das assinaturas dos protocolos estudados e análise das assinaturas obtidos.

Com esse estudo, foi concluído que a utilização do algoritmo LCS e nCS para extrair assinaturas afim de identificar os protocolos de aplicação, nestes testes, não foi muito eficaz, porque ele extrai uma assinatura extremamente pequena, que não garante a identificação de uma porta específica, pois em uma conexão de alta velocidade, essas assinaturas extraídas podem estar em todos os dados enviados e recebidos.

Esses algoritmos não é um algoritmo que calcula a maior subsequência comum de  $n$  seqüências (nLCS), o que é NP-Difícil, mas com base nessa idéia, espera-se estender o estudo sobre esse assunto, afim de promover melhores resultados e alcançando assim, o objetivo de maior segurança na rede.

## **7. Agradecimentos**

Primeiramente, agradeço pelo apoio financeiro (bolsa) concedido pela Universidade Estadual de Mato Grosso do Sul, aos professores que estão sempre ali dispostos a ajudar e aos meus colegas Bruno Cuencas Donath e Luana Kaku Aguiar, que me acompanhou e ajudou no desenvolvimento deste trabalho.

## **8. Referências Bibliográficas**

### **Artigos**

Izidre, S.L. **Identificação de Protocolos de Aplicação Através de Conteúdo** (Artigo em publicação). 2008

### **Livros**

Comer, Douglas E. **Interligação em rede com TCP/IP, Volume 1 – Princípios, protocolos e arquitetura**, 1998

Cormem, T. H., Leiserson C. E., Rivest R. L. e Stein C. **Introduction to Algorithms**. MIT Press, Cambridge, MA, 2001. Second Edition.

Gene Spafford e Simson Garfinkel. **Practical Unix & Internet Security**. O Reilly and Associates, Sebastopol, 2a edição, 1996.

Nakamura, E. e Geus P. **Segurança de redes em ambientes cooperativos**. Editora Futura, São Paulo, 3a edição, 2003.

#### **Sites da internet**

**Curso de redes de computadores – Internet e arquitetura TCP/IP, volume 1** – PUC RIO/CCE. Disponível on line na URL [www.rjunior.com.br/download/tcp.pdf](http://www.rjunior.com.br/download/tcp.pdf)

Jannuzzi, Fernanda S., **Protocolo TCP**. Disponível on line em junho de 2003 na URL [http://www.cbpf.br/~sun/pdf/tcp\\_apres.pdf](http://www.cbpf.br/~sun/pdf/tcp_apres.pdf)

**Rede Nacional de Ensino e Pesquisa. É vantajoso utilizar o protocolo IMAP**. Disponível on line em outubro de 1997 na URL <http://www.rnp.br/newsgen/9710/n5-2.html>

Ricarte, I. L. M., **HTTP**. Disponível on line em março de 2000 na URL <http://www.dca.fee.unicamp.br/cursos/PooJava/network/http.html>

Sandes, Edans F. O., **Secure Shell – Implementação de um cliente SSH**. Disponível on line em dezembro de 2003 na URL <http://www.cic.unb.br/docentes/pedro/trabs/SSH.htm>

#### **Teses e Dissertações**

Paula, F. S. **Uma arquitetura de segurança computacional inspirada no sistema imunológico**. Tese de doutorado, Instituto de Computação, Universidade Estadual de Campinas, 2004.