ANÁLISE DE TRÁFEGO QUALITATIVO (TCP/IP)

Anderson Alves de Albuquerque aaa@cbpf.br
Marita Maestrelli marita@cbpf.br

Outubro/2002

Prefácio

Para facilitar a depuração dos resultados obtidos através das ferramentas de IDS**[13]** , esta nota técnica abordará tópicos sobre análise de tráfego (TCP/IP) **[34]** em redes Ethernet (IEEE 802.3)**[7.1][14]** .

Após o entendimento sobre a análise de tráfego estaremos prontos para construir ferramentas e *scripts* de depuração deste tráfego.

Este documento também pretende ser a base sobre protocolos**[24]** para as próximas notas técnicas sobre segurança em redes..

Índice Geral:

1. Introdução

A característica favorável da abordagem por análise de tráfego é a possibilidade de examinar e registrar cada pacote que circula na rede. Situações onde a privacidade é um interesse crítico, como por exemplo, em instituições educacionais, a abordagem através da análise de tráfego pode ser a única forma razoável de monitoração da rede.

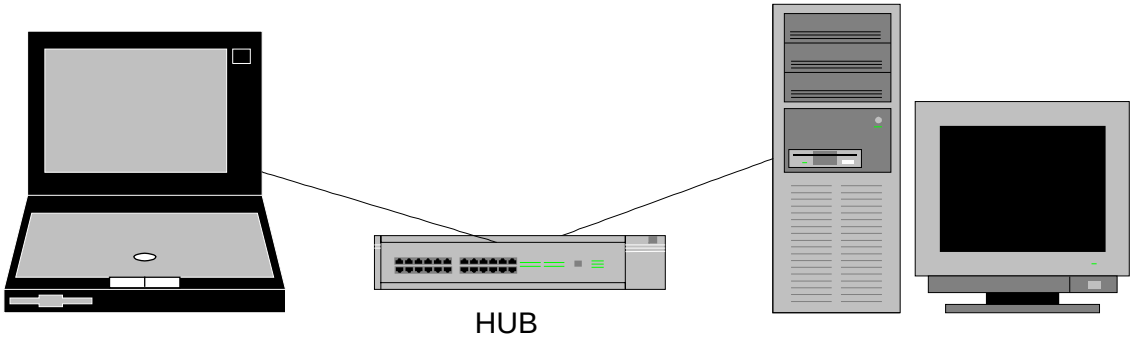
Nos ítens do capítulo 1 (um) faremos uma abordagem básica das funcionalidades das ferramentas no desenvolvimento deste trabalho, como: Sniffers**[30]**, RFC**[25]**, Linux/UNIX**[17]/[38]**, Scan de portas**[28]** .

Inicialmente, no capítulo 2, abordaremos os ensinamentos básicos sobre TCP/IP**[34]** como camadas e funcionalidade dos campos (ARP**[1]**, IP**[15]**,ICMP**[12]**, TCP**[33]**, UDP**[37]**) do TCP/IP**[34]** versão 4.

Após isso, iniciaremos a análise propriamente dita, sempre enfocando ção a segurança de redes e sistemas.

Outros tópicos serão abordados indiretamente como mudanças de base dos números. Usaremos bases binárias, hexadecimal**[9.1]** e decimal

1.1. Estrutura de rede usada para o estudo



Client:
IP = 10.10.10.2
Máscara de Rede: 255.0.0.0
MAC = 0:c0:26:10:16:bd
Hostname = CLIENT (Windows 98)

Server:
IP = 10.10.10.1
Máscara de Rede: 255.0.0.0
MAC = 0:e0:7d:85:f4:6a
Hostname= GW (Unix)

Acima está a estrutura de rede que foi usada para o estudo e coleta de dados.O servidor**[29]** 10.10.10.1 estava configurado com algumas ferramentas de ataque, scan de portas**[28]** e serviços básicos que serão abordados em testes, como: Telnet**[35.1]**, SSH**[31]**, FTP**[9]**, HTTP**[11]**, DNS**[5]**, POP3**[22]** e etc ...

1.2. Sniffer

São programas que tem a função de capturar os pacotes que trafegam no mesmo domínio de colisão onde esta ferramenta está instalada . A maioria dos sniffers**[30]** possuem filtros de protocolos**[24]** e/ou filtros de conteúdo.

No nosso caso usaremos o **TCPDump [35]**, alguns parâmetros úteis ao nosso estudo serão mostrados abaixo.

O TCPDump está presente na maioria dos linux**[17]** e Unix**[38]**, caso este não seja encontrado poderá ser facilmente obtido, compilado e instalado . Mas nada impede a utilização de outro sniffer**[30]** que tenha as mesmas propriedades do TCPDump.

O computador que fica monitorando a rede poderia ter os seus pinos de envio "Tx" no conector RJ-45 **[25.1]** desconectados "abertos". Como exemplo poderiamos usar a especificação 10Base-T (10 Mb**[18.3]**) do padrão IEEE 802.3, nesse caso não usariamos os pinos 1 e 2 que são os de envio "TX" . Isso tornaria o sistema de detecção de invasão mais "seguro" .

.

Autores: Van Jacobson, Craig Leres and Steven McCanne

Site: ftp://ftp.ee.lbl.gov/tcpdump.tar.Z

Parâmetros ilustrativos:

$ tcpdump port 23
(Filtra o tráfego da porta**[23]** 23)

$ tcpdump proto 6 and port 23
(Filtra o tráfego TCP**[33]** na porta**[23]** 23, os códigos de protocolos**[24]** podem ser listados em /etc/protocols)

$ tcpdump proto 1
(Filtra os pacotes icmp**[12]**)

$ tcpdump proto 17 and not port 53
(Filtra o tráfego UDP exceto o tráfego UDP**[37]** na porta**[23]** 53)

$ tcpdump -i rl0
(Seleciona a interface rl0)

$ tcpdump –x
(Filtra o tráfego mostrando o cabeçlho e corpo do pacote em Hexadecimal**[9.1]** e em ASCII)

Existem diversos parâmetros que podem ser listados no manpage**[18.2]** do TCPDump**[35]**, a listagem acima foi apenas para ilustrar a sua utilização .

1.3. RFC (Request for Comments)

As RFCs**[25]** são um conjunto de anotações informais que relatam história, arquitetura, protocolos**[24]** e serviços usados em redes (Internet). Esses documentos são ricos em informações e detalhes.

Inicialmente podemos obter a RFC 1000 e a RFC 1750 que possuem os índices sobre as RFCs anteriores. Com a RFC-index.txt, teremos a listagem completa.

Site, via HTTP[11]:

http://www.rfc-editor.org
http://www.cis.ohio-state.edu/cs/Services/rfc/rfc-text/

Site, Via FTP[9] anonymous (Guest):
ftp://ds.internic.net
ftp://ftp.isi.edu/in-notes/

1.4. Scan de portas[23]

Estas ferramentas enviam pela rede pacotes com campos pré-definidos. Os campos pré-definidos podem ser: portas**[23]**, protocolos**[24]**, flags**[8]**, conteúdo de dados, IP de origem, IP destino **[15]**e outros parâmetros .

No nosso projeto usamos o **nmap [18.5]**, onde **nmap** é uma ferramenta de exploração de rede e *scanner* de segurança.
Nmap foi projetado para permitir aos administradores de sistemas explorar grandes redes para determinar quais computadores estão ativos e quais serviços são fornecidos. Nmap suporta um grande número de técnicas de scan, como: UDP, TCP connect(), TCP SYN (half open), ftp proxy (bounce attack), Reverse-ident, ICMP (ping sweep), FIN, ACK sweep, Xmas Tree, SYN sweep, IP Protocol, and Null scan. Nmap, também, oferece outras características, como: detecção remota do SO via TCP/IP fingerprinting, stealth scanning, dynamic delay e retransmission calculations, scanning paralelo, detecção de hosts inativos através de pings paralelos, decoy scanning, detecção de portas filtradas, scanning direto de RPC (não portmapper), fragmentation scanning e flexibilidade do alvo e especificação de porta.

Autor: Fyodor <fyodor@dhp.com>

Site: http://www.insecure.org/nmap/

Exemplos:

$ nmap -p 23 10.10.10.1
(Envia para a porta**[23]** 23 um pacote TCP)

$ nmap -p 1-1023 10.10.10.1
(Envia para as portas**[23]** 1 à 1023 pacotes TCP)

$ nmap -p 23 –sS 10.10.10.1
(Envia para a porta**[23]** 23 um pacote TCP com o flag Syn acionado)

$ nmap -p 23 –sR 10.10.10.1
(Envia para a porta**[23]** 23 um pacote TCP com o flag Reset acionado)

$ nmap –p 23 –sU 10.10.10.1
(Envia para a porta**[23]** 23 um pacote UDP)

1.5. Scan de Vulnerabilidades

Devido ao grande número de vulnerabilidades reportadas todos os dias em listas de discussão e sites de segurança, torna-se muito difícil para administradores de redes efetuarem todos os testes necessários para assegurar a integridade de seus hosts. Para isso, surgem os softwares de detecção de vulnerabilidades.

Nessus é um scanner de segurança/vulnerabilidades remoto para Linux, BSD, Solaris e outros *nix em geral. Nessus é uma ferramenta para auditoria de segurança. Com ele é possível verificar varias vulnerabilidades em sua rede. Trabalha com multi-threading e é baseado em plugins, além de possuir uma interface GTK que facilita sua configuração/entendimento e tem a capacidade de fazer mais de 500 checks remotos de segurança. Permite a exibição de um relatório gerado em HTML, XML, LaTeX e texto ASCII, além de sugerir uma solução para cada problema encontrado.

Autor: Renaud Deraison

Site: http://www.nessus.org

1.6. Valores de protocolos[24] e portas[23]

Em diversos momentos precisaremos de informações básicas como número de portas**[23]** e códigos protocolos**[24]**, e para tal devemos consultar os arquivos existente nos sistemas operacionais:

Linux**[17]** e Unix**[38]**:

Arquivos	Conteúdo
/etc/services	Relação de portas e serviços
/etc/protocols	Relação de código e protocolos

Windows 9.X**[40]**:

Arquivos	Conteúdo
X:\windows\services	Relação de portas e serviços
X:\windows\protocol	Relação de código e protocolos

OBS: Estas informações também podem ser encontradas nas RFCs **[25]**.

2. Revisão sobre TCP/IP

Metodologia usada:

Os valores em Hexadecimal**[9.1]** dos campos estão alternados em negrito ou não, para facilitar a visualização. No final, os valores em itálicos são os dados.

Quando necessário, a base numérica será representada por "(x)" subscrito, onde x é a base numérica .

Qualquer dúvida sobre a utilização do TCPDump**[35]** poderá ser vista no manpage do TCPDump**[35]**.

Camadas ISO/OSI**[16]**com os protocolos**[24]** que serão abordados abaixo:

Aplicação	
Apresentação	
Sessão	
Transporte	(TCP e UDP)
Rede	(IP, ICMP)
Enlace	(ARP)
Hardware	

2.1. Protocolo ARP (Address Resolution Protocol)

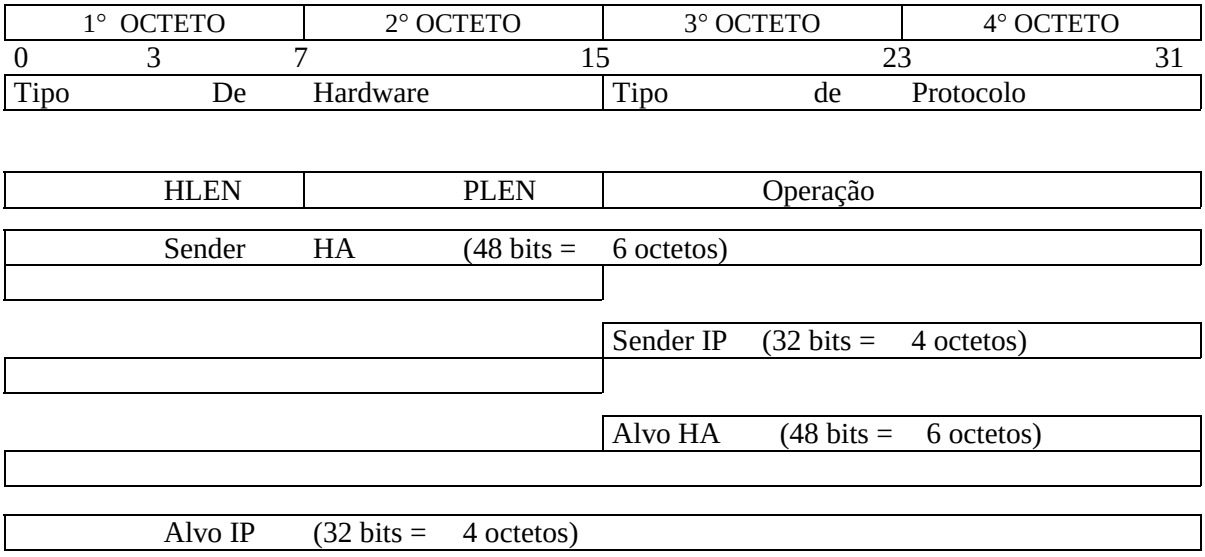
O ARP**[1]** é usado para mapear o endereço físico das entidades envolvidas na conexão. O ARP**[1]** usa o endereço físico contido na ROM**[26]** da interface de rede, no caso das redes Ethernets**[7.1],** o MAC (Medium Access Control)**[18]** possuí 48 bits**[2]** (24 bits reservado ao fabricante e os 24 finais reservados a placa). A notação usada pelo endereço de MAC (Medium Access Control)**[18]** são 6 (seis) octetos**[20]** escritos em Hexadecimal**[9.1]** separados por dois pontos.

Exemplo: 00:C1:A1:7C:12:AA

Esse endereço físico é pertinente a tecnologia de rede usada, ou seja, caso o datagrama tenha que passar de uma rede ethernet**[7.1]** para uma X25**[41]** o roteador terá que remover as informações de enlace do ethernet**[7.1]** e anexar as informações de enlace de X25**[41]**. Quando houver outra passagem de uma rede de tecnologia "X" para "Y" o mesmo processo será feito . Quando um datagrama vai de um roteador "X" para "Y", o MAC "X" é colocado no campo de origem e o MAC "Y" é colocado no destino. Quando o roteador "Y" enviar o datagrama para o roteador "Z", o MAC de "Y"será colocado como endereço físico de origem e o MAC "X" no endereço de destino. Após estes processos o datagrama é enviado pela rede.

Este endereço físico é "imutável" na placa de rede através de softwares, porém não podemos excluir a possibilidade deste ser modificado no driver de placa de rede na camada de enlace por programas maliciosos.

Em particular existe um programa deste tipo muito usado no sistema operacional windows**[40]** 9x e NT .



- **Exemplo Prático do Protocolo ARP:**

No server**[29]** é usado um ping**[21]** para o endereço inexistente 10.10.10.222 .

```
$ tcpdump -x -e -n  not port 23 and not port 22
14:19:21.211893 0:e0:7d:85:f4:6a ff:ff:ff:ff:ff:ff 0806 60: arp who-has 10.10.10.222 tell
10.10.10.1
      0001 0800 0604 0001 00e0 7d85 f46a 0a0a
      0a01 0000 0000 0000 0a0a 0ade                    (Cabeçalho ARP)

     0000 0000 0000 0000 0000 0000 0000 0000 0000     (Dados ARP)
```

- ## Análise qualitativa dos campos

0001$_{(16)}$	Tipo de Hardware (código 0001$_{(16)}$ usado para ethernet); Token ring = 0004$_{(16)}$
0800$_{(16)}$	Protocolo usado no nível superior (0800$_{(16)}$ corresponde ao IP); 0800$_{(16)}$ = 2048$_{(10)}$ Apple talk: 8037$_{(16)}$ = 32823$_{(10)}$ Frame Relay: 000F$_{(16)}$ = 15$_{(10)}$
06$_{(16)}$	Extensão do endereço de Hardware;
04$_{(16)}$	Extensão do protocolo de alto nível;
0001$_{(16)}$	Operação executada Protocolo; 0001 – Arp Ask; 0002 – Arp answer; 0003 – Rarp ask; 0004 – Rarp answer;
00e07d85f46a$_{(16)}$	Mac da origem;
a0a0a0a01$_{(16)}$	IP de origem, onde: 0a 0a 0a 01$_{(16)}$ = 10.10.10.1
0000 0000 0000$_{(16)}$	Arp do destino, como este ainda não é conhecido este deve ser zero;
0a 0a 0a de$_{(16)}$	IP destino, onde: 0a 0a 0a de$_{(16)}$ = 10.10.10 . 222

- ## Explicação sobre os campos relevantes

- Supondo código de operação 1 (Arp Ask) as variáveis conhecidas são: IP Origem, IP destino, MAC**[18]** origem. Neste caso apenas o MAC**[18]** destino é desconhecido.

Neste caso estamos solicitando o MAC**[18]** address de algum Host**[10]** para podermos fechar um enlace.

- Supondo código de operação 2 (Arp Answer) as variáveis conhecidas são: IP de origem**[15]**, IP destino**[15]**, MAC**[18]** de origem e MAC**[18]** destino.

Neste caso estamos respondendo a um pedido de código de operação 1, informando o MAC**[18]** address do nosso Host**[10]** para que o outro Host**[10]** efetue um enlace e transmissão do datagrama .

- Supondo código de operação 3 (RARP ask) as variáveis conhecidas são: MAC de origem e MAC destino**[18]**.

Neste caso uma solicitação de Bootp solicita o endereço ip de origem na inicialização do computador, há uma resposta em seguida pelo código de operação 4.

- Supondo código de operação 4 (RARP answer) as variáveis conhecidas são: IP origem, MAC de origem**[18]** e MAC destino**[18]**.

Neste caso uma solicitação de Bootp o IP é dado pelo servidor**[29]**, de certa forma o IP de origem é "conhecido" ou imposto ao receptor.

Algumas RFCs podem ser consultadas, como: 903, 1293, 826, 814 .

2.2. Protocolo IP (Internet Protocol)

O IP (Internet Protocol)**[15]** é um protocolo**[24]** que consiste em um mecanismo de entrega de pacote fim-a-fim e não-confiável. O IP**[15]** especifica o tamanho do pacote além de ser responsável pelo roteamento, e também faz verificações de integridade do pacote. Podemos dizer até que o IP**[15]** é a unidade básica do TCP/IP**[34]** .

O IP**[15]** tem duas funções básicas:

implementar método de acesso a rede e
fragmentação .

Dois campos do IP**[15]** definem o IP**[15]** de origem (32 Bits) **[15]** e IP de destino (32 Bits**[2]**) **[15]**, endereço este separados por 4 octetos**[20]** escrito em base decimal .

<table>
<tr><td colspan="2">1° OCTETO</td><td colspan="2">2° OCTETO</td><td colspan="2">3° OCTETO</td><td colspan="2">4° OCTETO</td></tr>
<tr><td>0</td><td>3</td><td>7</td><td>11</td><td>15</td><td>19</td><td>23</td><td>27</td><td>31</td></tr>
<tr><td colspan="2">Versão</td><td>Hlen</td><td colspan="2">Tipo Serviço</td><td colspan="4">Tamanho Total</td></tr>
<tr><td colspan="4">Identifica ção</td><td>Flags</td><td colspan="3">Frag OffSet</td></tr>
<tr><td colspan="2">Tempo de vida</td><td colspan="2">Protocolo</td><td colspan="5">Check Sum (Cabeçalh o)</td></tr>
<tr><td colspan="9">IP de Origem (4 Octetos = 32 Bits)</td></tr>
<tr><td colspan="9">IP de Destino (4 Octetos = 32 Bits)</td></tr>
<tr><td colspan="7">Opcional (Se houver)</td><td colspan="2">Padding</td></tr>
<tr><td colspan="9">Data ...</td></tr>
</table>

- ## Exemplo Prático do Protocolo IP:

No cliente**[3.2]** é usado uma conexão TCP**[33]** para o endereço do Server**[29]** 10.10.10.2 porta**[23]** 80.

```
$ tcpdump -x -n -e port 80
19:17:39.784173 0:c0:26:10:16:bd 0:e0:7d:85:f4:6a 0800 62: 10.10.10.2.1031 >
10.10.10.1.80: S 10960869:10960869(0) win 8192 <mss 1460,nop,nop,sackOK>
(DF)

 4500 0030 9704 4000 8006 3bad 0a0a 0a02  0a0a 0a01          (IP)

 0407 0050 00a7 3fe5 0000 0000 7002 2000 f625 0000
0204 05b4 0101 0402                                        (TCP)
```

- ## Analise qualitativa dos campos:

4$_{(16)}$	Versão do IP;
5$_{(16)}$	Tamanho do cabeçalho em unidades de 32 Bits, sendo o tamanho mínimo igual a 5;
00$_{(16)}$	TOS, define o tratamento do datagrama ao longo do caminho;
0030$_{(16)}$	Tamanho total em 8 bits (Octetos), todo o pacote IP + TCP. Valor máximo = 65535 octetos. Para o ethernet valor Máx.=1500 octetos e para FDDI=4470 octetos ; 0030$_{(16)}$ = 48$_{(10)}$ Octetos.
9704$_{(16)}$	Identificação do Datagrama, ajuda na remontagem;
4$_{(16)}$	Flags. Indica Fragmentação ou não.
000$_{(16)}$	OFFSet, Deslocamento do Fragmento. O primeiro fragmento é indexado com zero;
80$_{(16)}$	Tempo de vida (TTL), quando o decrementa até 0 (Zero) é destruído e retornado um pacote de erro ;
06$_{(16)}$	Protocolo de nível superior;
3bad$_{(16)}$	CheckSum = 11101110101101$_{(2)}$ Usado pelo ethernet o CRC-32 ;
0a0a 0a02$_{(16)}$	IP origem = 10.10.10.2 ;
0a0a 0a01$_{(16)}$	IP destino = 10.10.10.1 .
-	Padding é usado para que o pacote seja múltiplo de 32 bits em função do campo “Hlen” .

- **Explicação sobre os campos relevantes**

Campo Flag*[8]*:

0	1	2
Reservado	DF (Não fragmento)	MF (Mais fragmento)

MF	O	Na fragmentação esse bit identifica o último fragmento.
	1	Na fragmentação indica que o fragmento não é o último.

DF	Define se o pacote pode ou não ser fragmentado, caso no meio do caminho precise fragmentar o roteador envia um mensagem de erro descartando o datagrama original.

Campo CheckSum (CRC):

Polinômio usado no campo de CheckuSum**[3]** para verificar a integridade do datagrama IP é o:

$$CRC\text{-}32 = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+ x + 1$$

Campo TOS (Tipo de Serviço) :

0	1	2	3	4	5	6	7
Precedência	Precedência	Precedência	D	T	R	Novo	Novo

Nos campos de Precedência a prioridade crescente de acordo com os valores setados nos 3 Bits**[2]**.

O campo D solicita baixo intervalo, T solicita uma banda maior, R solicita maior confiabilidade.

OBS: O campo *Options* indica tipo de pacote (dado ou controle). Além disso informações e opções, como: Aviso sobre erros gerais, TimeStamp, Loose Source Routing, Strict Source Routing e Record Route.

Algumas RFCs podem ser consultadas, como: 791, 815, 894, 1122 e 1700.

2.3. Protocolo ICMP (Internet Control Message Protocol)

O ICMP**[12]** tem a função de controlar ou informar sobre possíveis erros que podem ocorrer, por exemplo:

Host**[10]** ou rede inalcançáveis,

Rede com a banda ocupada,

porta**[23]** UDP**[37]** fechada,
verificação se um determinado IP está ativo,
erros de protocolo**[24]**,
mostrar por quais roteadores o pacote passa,
solicitação de sincronismo e
solicitação de mascaras de rede (este atua na camada de rede).

Apesar de parecer simples o protocolo**[24]** ICMP**[12]** deve ser estudado pois muitas backdoors são implementadas sobre o ICMP**[12]** adicionando comandos no corpo do pacote ICMP**[12]**. Nestas backdoors são enviados comandos no corpo através de um simples pacote echo**[7]** request gerado por um pseudo-ping**[21]**. Quando o pacote chega na rede parece simplesmente um "ping", mas o comando malicioso está no corpo do ICMP**[12]** .

Em um firewall devemos ter cuidado ao bloquear o comando do tipo iguais à 3, 4, 5 e 12 . Porém os outros podem ser bloqueados a princípio, Ex: 0 (echo reply) **[7]**, 8 (echo request), 11 (TTL), 13, 14, 15, 16, 17 e 18 . No caso de um roteador o valor do campo tipo 13 e 14 pode ser interessante não bloquear através de filtros.

1° OCTETO	2° OCTETO	3° OCTETO	4° OCTETO
0 3 7	11 15	19 23	27 31
Tipo	Código	Check Sum (icmp)	
Identificador		Número de Sequência	
Dados Opcionais ...			

- **Exemplo Prático do Protocolo ICMP:**

No server**[10]** é usado um ping**[21]**para o endereço do cliente**[3.2]** 10.10.10.2 .

```
$ tcpdump -x -e -n  proto 1
19:28:12.762723 0:c0:26:10:16:bd 0:e0:7d:85:f4:6a 0800 74: 10.10.10.2 > 10.10.10.1:
icmp: echo request
4500 003c f805 0000 2001 7aa5 0a0a 0a02 0a0a 0a01          (IP)

0800 4b5c 0100 0100                                        (ICMP Request)
6162 6364 6566 6768 696a 6b6c 6d6e 6f70 7172 7374 7576 7761 6263 6465 6667 6869

19:28:12.762969 0:e0:7d:85:f4:6a 0:c0:26:10:16:bd 0800 74: 10.10.10.1 > 10.10.10.2:
icmp: echo reply

4500 003c 0583 0000 ff01 8e27 0a0a 0a01 0a0a 0a02          (IP)

0000 535c 0100 0100                                        (ICMP Reply)
6162 6364 6566 6768 696a 6b6c 6d6e 6f70 7172 7374 7576 7761 6263 6465 6667 6869
```

- ## Analise qualitativa dos campos:

08$_{(16)}$	Tipo, 08$_{(16)}$ é Echo Request e 00$_{(16)}$ é echo Reply ;
00$_{(16)}$	Código;
4b5c$_{(16)}$	CheckSum = 100101101011100 $_{(2)}$;
0100$_{(16)}$	Identificador;
0100$_{(16)}$	Número de sequência;

- ## Explicação sobre os campos relevantes

Os pacotes ICMP**[12]** sempre tem os 3 primeiros campos como obrigatório (Tipo, código e checksum**[3]**).

O campo tipo fornece informações sobre a mensagem e o campo código tem informações adicionais. Basicamente o campo tipo mostra qual o erro ou solicitação (echo, reply, destino inalcançável e etc ...) já o código informa onde ocorreu o erro (Host**[10]**, Rede, Roteador, protocolo**[24]** e etc ...) .

Resumidamente, o campo tipo reponde a pergunta “O que está acontecendo?” e o campo código responde “Onde está acontecendo?” .

O checksum**[3]** verifica a integridade do pacote transmitido.

O campo de identificador e número de seqüência combina a resposta com o pacote original, mas nem sempre são usados.

De acordo com o campo tipo, podemos apresentar o formato do protocolo**[24]** ICMP**[12]**. Abaixo são mostradas as informações que aparecerão nos diversos campos:

Tipo	Nome	Uso
0	Echo Reply	Resposta a solicitações
3	unreachable	Destino não está acessível
4	Quench	Controle de congestionamento.
5	Rota	Mudança de rota
8	Echo Request	Solicitação de resposta
11	TTL	Tempo de vida
12	Err	Relato de erro não coberto pelo ICMP
13	Time Request	Pedido de sincronismo
14	Time Reply	Resposta a sincronismo
17	Mask Request	Solicitação da máscara
18	Mask Reply	Resposta a solicitação máscara

Tipo	Identificador e seqüência	Dados
0	✔	Opcionais
3	Não usado e o conteúdo deve ser zero	Internet Header e os 64 bits da solicitação que original a mensagem
4	Não usado e o conteúdo deve ser zero	Internet Header e os 64 bits da solicitação que original a mensagem
5	Endereço do roteado	Internet Header e os 64 bits da solicitação que original a mensagem
8	✔	Opcionais
11	Não usado e o conteúdo deve ser zero	Internet Header e os 64 bits da solicitação que original a mensagem
12	Ponteiro para o octeto que ocasionou o erro (8 bits), os 24 bits restantes são zero	Internet Header e os 64 bits da solicitação que original a mensagem
13	✔	Informação sobre sincronismo
14	✔	Informação sobre sincronismo
17	✔	Máscara
18	✔	Máscara

✔ - Existente

Tipo	Código
0	Não setado (0)
3	0 = rede não encontrada; 1 = host não encontrado; 2 = problema com protocolo; 3 = porta não encontrada; 4 = fragmentação necessária.
4	Não setado (0)
5	0 = redirecionar para a rede; 1 = redirecionar para o host; 2 = Redirecionar para tipo de serviço e rede; 3 = Redirecionar para tipo de serviço e Host.
8	Não setado (0)
11	0 = TTL; 1 = TTL exedido.
12	0 = problema com opções.
13	Não setado (0)
14	Não setado (0)
17	Não setado (0)
18	Não setado (0)

OBS: Quando ocorre erro de transmissão o pacote retorna com o cabeçalho (Internet Header) + 64 Bits (8 octetos) da mensagem que ocasionou o erro .

OBS: No cabeçalho IP do exemplo mostrado acima o tamanho total em $003c_{(16)} = 60_{(10)}$ octetos corresponde ao tamanho total do pacote.

Algumas RFCs podem ser consultadas, como: 777, 792, 896, 950, 956, 957, 1016, 1122, 1256, 1305 e 1931.

RFCs ICMP V6: 1885, 2463 e 2466 .

2.4. Protocolo UDP (User Datagram Protocol)

O UDP**[37]** atua na camada de transporte e é um protocolo**[24]** não confiável, onde o projetista do serviço deve assumir a possibilidade de eventuais erros. O UDP não verifica ordem de chegada, banda disponível na rede, não verifica eventuais erros e não possui recursos de controle de erro e check.

Podemos notar que os projetista e programadores devem tomar muito cuidado na utilização do UDP .

Devida a sua facilidade e simplicidade que será vista nos campos em comparação ao TCP, o UDP é mais rápido e menor.

Alguns serviços que utilizam o UDP:
Echo (7)**[7]**, Whois**[39]** (43), Domain (53)**[6][5]**, TFTP (69)**[36]**, NTP (123)**[19]**, SunRPC (111) **[32]** e outros .

Para acessar as portas**[23]** o UDP usa o mecanismo de desmultiplexação e multiplexação de portas**[23]** e o número de portas**[23]** pode chegar à 65535 como será visto abaixo.

1° OCTETO		2° OCTETO		3° OCTETO		4° OCTETO	
0	3	7	11	15	19	23	27	31
Porta Origem				Porta Destino			
Tamanho da mensa gem				Check Sum (UDP)			
Dados							

- **Exemplo Prático do Protocolo UDP:**

No server**[29]** é usado um pacote udp para o endereço do cliente**[3.2]** 10.10.10.2 porta**[23]** 81 .

```
$ tcpdump -x -e -n   port 81
19:39:29.633810 0:e0:7d:85:f4:6a 0:c0:26:10:16:bd 0800 60: 10.10.10.1.48035 >
10.10.10.2.81:  udp 0
4500 001c b5fa 0000 3a11 a2c0 0a0a 0a01  0a0a 0a02                (IP)

bba3 0051 0008 1bd3                                              (UDP)
0000 0000 0000 0000 0000 0000 0000 0000 0000
```

- **Análise qualitativa dos campos:**

Bba3$_{(16)}$	Porta Origem, porta = 48035 $_{(10)}$;
0051$_{(16)}$	Porta destino, porta = 81$_{(10)}$;
0008$_{(16)}$	Comprimento do cabeçalho em 8 bits (Octeto) por unidade;
1bd3$_{(16)}$	CheckSum = 1101111010011$_{(2)}$ (Opcional)

- **Explicação sobre os campos relevantes**

OBS: No UDP**[37]** o checksum**[3]** é feito com adição de um pseudo-cabeçalho: IP origem + IP destino + 8 bits em zero + Código do protocolo UDP (17$_{(10)}$) + comprimento UDP .

OBS: No cabeçalho IP o tamanho total em 001c$_{(16)}$ = 28$_{(10)}$ octetos**[20]** corresponde ao tamanho total do pacote.

Algumas RFCs podem ser consultadas, como: 768

2.5. Protocolo TCP (Transmission Control Protocol)

O TCP**[33]** é um protocolo**[24]** orientado a conexão e confiável na camada de transporte, usado por diversos serviços. Assim como o UDP**[37]** há multiplexação e desmultiplexação de portas**[23]**, no total teremos 65535 portas**[23]** .

As principais características do TCP**[33]** são: Circuito virtual, *stream* de dados, transmissão com *buffers* e conexão *full duplex*.

Na transmissão do TCP**[33]** eventuais erros serão retransmitidos, o TCP também exige que as aplicações executem um procedimento de abertura e fechamento de uma conexão.

Um ponto importante do TCP são as janelas deslizantes, responsáveis pela negociação do tamanho dos pacotes que iremos receber.

1° OCTETO			2° OCTETO			3° OCTETO			4° OCTETO		
0	3	7	9		15	19		23	27		31
Porta de Origem						Porta de Destino					
Número de Sequência											
Número de Reconhecimento											
HLEN	Reservado		Flags			Janela					
CheckSum						Ponteiro Urgente					
Opções									Padding		
Dados ...											

- **Exemplo Prático do Protocolo TCP:**

No cliente**[3.2]** é usado uma conexão TCP para o endereço do Server**[10]** 10.10.10.1 porta**[23]** 80.

```
$ tcpdump -x -n -e port 80
19:17:39.784173 0:c0:26:10:16:bd 0:e0:7d:85:f4:6a 0800 62: 10.10.10.2.1031 >
10.10.10.1.80: S 10960869:10960869(0) win 8192 <mss 1460,nop,nop,sackOK> (DF)
 4500 0030 9704 4000 8006 3bad 0a0a 0a02 0a0a 0a01          (IP)

 0407 0050 00a7 3fe5 0000 0000 7002 2000 f625 0000 0204 05b4   (TCP)
 0101 0402
```

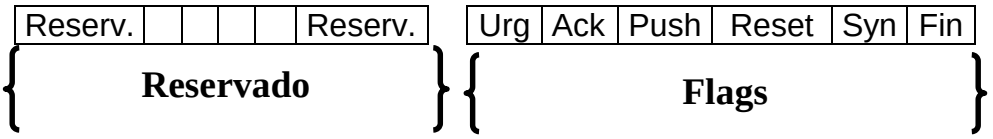
- **Analise qualitativa dos campos:**

0470$_{(16)}$	Porta Origem, Porta = 1031$_{(10)}$
0050$_{(16)}$	Porta Destino, Porta = 80$_{(10)}$
00a7 3fe5$_{(16)}$	Número de Sequência
0000 0000$_{(16)}$	Número de Reconhecimento. Confirmações de pacotes recebidos.
7$_{(16)}$	Tamanho do cabeçalho em unidades de 32 Bits (4 octetos)
002$_{(16)}$	Reservado + Código Bit, Neste caso foi acionado um Syn. Caso fosse um ACK+SYn do handshake = 012 Caso fosse um ACK seria = 010
2000$_{(16)}$	**Janela, win = 8192 $_{(10)}$ pacotes, variável com o tempo .**
f625$_{(16)}$	CheckSum = 1111011000100101$_{(2)}$
0000$_{(16)}$	**Ponteiro Urgente. Especifica a posição onde os dados urgentes terminam quando URG é inicializado nos Flags.**
0204 05$_{(16)}$	Opções, Negocia o MSS evitando que o buffer TCP fique cheio. Possibilita facilidades adicionais para usos futuros.
B4$_{(16)}$	Padding, enchimento para o pacote ser múltiplo de 32 Bits.

- **Explicação sobre os campos relevantes**

Campo Reservado: Para utilizações futuras.

Bits**[2]** Códigos: Reservado + Flags (Códigos bits):



Bits	Significado
Urg	Válido quando o “ponteiro urgente é valido”;
Ack	Confirmação;
Push	O receptor deve passar os dados o mais rápido possível;
Reset	Reinicializa uma conexão, interrompida de repente;
Syn	Pedido de sincronia. Exemplo: no handshake ;
Fin	Finalização de uma conversa. Ex: finalização elegante.

Campo Ponteiro Urgente: É um ponteiro para a posição onde os dados urgentes terminam, necessário em aplicações de login remoto . Com isso esse pacote deve passar a frente dos outros na fila pois é dada prioridade a este.

Janelas deslizantes: A aplicação de janelas mais simples é a janela de tamanho 1 (um) onde é necessário receber a confirmação do primeiro pacote para enviar o segundo e assim por diante, é óbvio a falta de rendimento deste esquema de confirmação. Foi criado então a janela deslizante, onde o tamanho da janela é definido de acordo com a banda e disponibilidade dos extremos. Depois disso são enviados os “N” pacotes definidos na janela, aguarda–se a confirmação para a janela deslizar. Mas, se houver a recepção dos pacotes “1”, “2”, “N-x”, “N-x+1” a janela pode deslizar até a posição “2” caso “N-x” seja diferente de 3. Quando o pacote “3” e “4“ forem confirmados o deslizamento continua assim por diante de forma sequencial até o e-nésimo pacote. Quando o range de 1 a “N” pacotes for transmitido teremos ranges subsequentes, e o processo continua até transmitir toda a informação desejada.

Campo Opções: O campo de opções serve por exemplo para negociar o MSS (Maximum Segment Size) negociado entre as extremidades .

Valor	Length	Uso
0	-	Final da lista de opções;
1	-	-
2	4	MSS .

Valor = 2	Length = 4	MSS com	16 Bits (2 Octetos)	Padding
00000010	00000100			XXXXXXXX

MSS

OBS: No TCP o checksum**[3]** é feito com adição de um pseudo-cabeçalho: IP origem + IP destino + 8 bits em zero + Código do protocolo TCP ($6_{(10)}$) + Comprimento TCP . Este pseudo-cabeçalho totaliza 96 Bits .

Algumas RFCs podem ser consultadas, como: 793, 813, 879 e 1122 .

3. Análise de tráfego & Teoria

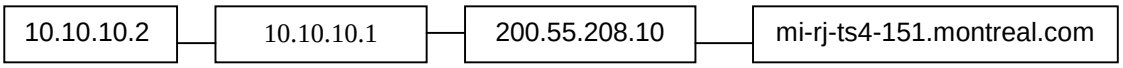
- **ICMP[12] Echo (8) e Reply (0):** Ping**[21]** para 10.10.10.2 .

```
05:17:37.324356 10.10.10.2 > 10.10.10.1: icmp: echo request
                4500 003c b60a 0000 2001 bca0 0a0a 0a02
                0a0a 0a01 0800 4a5c 0100 0200 6162 6364
                6566 6768 696a 6b6c 6d6e 6f70 7172 7374
                7576 7761 6263 6465 6667 6869

05:17:37.324624 10.10.10.1 > 10.10.10.2: icmp: echo reply
                4500 003c 00ce 0000 ff01 92dc 0a0a 0a01
                0a0a 0a02 0000 525c 0100 0200 6162 6364
                6566 6768 696a 6b6c 6d6e 6f70 7172 7374
                7576 7761 6263 6465 6667 6869
```

Podemos observar neste o envio de ICMP**[12]** tipo Echo (8) e recebimento de ICMP**[12]** tipo reply (0), de 10.10.10.2 para 10.10.10.1 .

- **ICMP[12] - destino inalcançável (3):**
  Ping**[21]** para mi-rj-ts4-151.montreal.com




```
14:58:01.088  10.10.10.2  > mi-rj-ts4-151.montreal.com.br: icmp: echo request

14:58:01.215  200.55.208.10  >  10.10.10.2:icmp:host   mi-rj-ts4-151.montreal.com
unreachable
```

```
14:58:02.224  10.10.10.2  > mi-rj-ts4-151.montreal.com: icmp: echo request

14:58:02.375 200.55.208.10  >  10.10.10.2: icmp: host mi-rj-ts4-151.montreal.com
unreachable
```

Podemos ver o recebimento de um icmp**[12]** tipo (3) enviado por mi-rj-ts4-151.montreal.com informando que o *host***[10]** destino não está na rede. O envio através de 10.10.10.2 é feito pelo icmp**[12]** tipo (8) .

- **ICMP[12] – Traceroute (11):** Traceroute**[36.2]** para 200.55.208.9, apresentando alguns códigos de problemas.

```
 $ traceroute to 200.55.208.9 (200.55.208.9), 30 hops max, 40 byte packets
 1    termsrv3.montreal.com (200.55.208.10)  137.348 ms  126.370 ms  126.884
ms
 2    termsrv4.montreal.com (200.55.208.9)  126.892 ms !H  127.264 ms !H
127.795 ms !H
```

!H	Host com problemas	!P	Protocolo com Problemas	!F	Falha em Fragmento
!N	Rede com problemas	!S	Falha em Source Route	!X	Comunicação administrativamente c/ problemas

Neste caso podemos observar o traceroute**[36.2]** em ação retornando alguns erros de controle, neste caso o erro foi mostrado por (!H) .

- **ICMP[12] – Traceroute (11):** Traceroute**[36.2]** para 200.55.208.1 , o sniffer**[30]** em 10.10.10.1 na interface interna diretamente conectada a client.ax3.com.br (10.10.10.2) .

```
c:\>  traceroute 200.55.208.1
traceroute to 200.55.208.1 (200.55.208.1), 30 hops max, 40 byte packets
 1  termsrv3.montreal.com  (200.55.208.10)  135.212 ms  136.032 ms  147.946 ms
 2  router.montreal.com  (200.55.208.1)  137.027 ms  137.350 ms  137.340 ms
```

```
15:07:56.859175 10.10.10.2 > router.montreal.com : icmp: echo request [ttl 1]
15:07:56.859480 10.10.10.1 > 10.10.10.2: icmp: time exceeded in-transit
15:07:56.864387 10.10.10.2 > router.montreal.com : icmp: echo request [ttl 1]
15:07:56.864599 10.10.10.1  > 10.10.10.2: icmp: time exceeded in-transit
15:07:56.866870 10.10.10.2 > router.montreal.com : icmp: echo request [ttl 1]
15:07:56.867053 10.10.10.1  > 10.10.10.2: icmp: time exceeded in-transit
15:07:57.883812 10.10.10.2 > router.montreal.com : icmp: echo request
15:07:58.064301 termsrv3.montreal.com  > 10.10.10.2: icmp: time exceeded
in-transit
```

```
15:07:58.066998 10.10.10.2 > router.montreal.com : icmp: echo request
15:07:58.204325 termsrv3.montreal.com  > 10.10.10.2: icmp: time exceeded in-transit
15:07:58.207158 10.10.10.2 > router.montreal.com : icmp: echo request
15:07:58.334316 termsrv3.montreal.com  > 10.10.10.2: icmp: time exceeded in-transit
15:07:59.227096 10.10.10.2 > router.montreal.com : icmp: echo request
15:07:59.364381 router.montreal.com  > 10.10.10.2: icmp: echo reply
15:07:59.389745 10.10.10.2 > router.montreal.com : icmp: echo request
15:07:59.534384 router.montreal.com  > 10.10.10.2: icmp: echo reply
15:07:59.536544 10.10.10.2 > router.montreal.com : icmp: echo request
15:07:59.674432 router.montreal.com  > 10.10.10.2: icmp: echo reply
```

Neste caso podemos observar o traceroute**[36.2]** em ação enviando icmp**[12]** tipo (8) com TTL aumentando gradativamente em x++, o acréscimo de TTL++ é feito quando o TTL é decrescido pelos roteadores e retornar a valor 0 (zero). Em seguida é enviado outro icmp**[12]** tipo (8) para o próximo roteador com TTL acrescido de 1 (uma unidade) relativa ao último envio decrescido até zero.

- **DNS: Consulta recursiva e não-reversa:** Consulta feita por 10.10.10.2 sendo o dns**[5]** primário o 10.10.10.1 , procurando pelo sever www.uff.br .

```
15:29:10.269473 10.10.10.2.1115 > 10.10.10.1.53:  1+ A? www.uff.br. (28)
15:29:11.793107 10.10.10.2.1115 > 10.10.10.1.53:  1+ A? www.uff.br. (28)
15:29:12.516260 10.10.10.1.53 > 10.10.10.2.1115:  1 1/0/0 A www.uff.br (44)
15:29:12.736015 10.10.10.1.53 > 10.10.10.2.1115:  1 1/0/0 A www.uff.br (44)
```

Neste caso podemos ver o cliente**[3.2]** 10.10.10.2 enviando requisições udp para a porta**[23]** 53 do seu server de dns**[5]** (10.10.10.1). Esta requisição tem o intuito de resolver o *hostname***[10]** www.uff.br . Vale ressaltar que o cliente**[3.2]** usa a porta**[23]** 1115 que é maior do que 1023, conforme as recomendações .

- **Handshake (3-Way):** Conexão na porta**[23]** 80 do *server* 10.10.10.1.

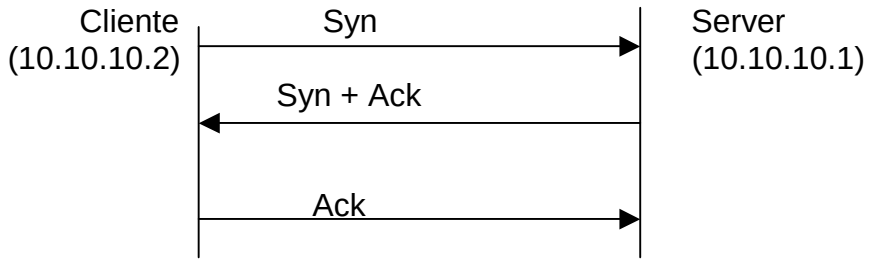
```
16:20:06.377499 10.10.10.2.1144 > 10.10.10.1.80: S 9324403:9324403(0) win 8192 <mss 1460,nop,nop,sackOK> (DF)

16:20:06.378189 10.10.10.1.80 > 10.10.10.2.1144: S 61735258:61735258(0) ack 9324404 win 17520 <mss 1460> (DF)

16:20:06.379099 10.10.10.2.1144 > 10.10.10.1.80: . ack 1 win 8760 (DF)
```

Curiosidade: Neste exemplo podemos notar que o campo *win* é variável, cliente**[3.2]** com porta**[23]** maior que 1023 conforme recomendações.

Neste caso estamos mostrando o *handshake* de 3 vias iniciando uma sessão de TCP.



- **Finalização de maneira elegante:** Sessão FTP**[9]** do client 10.10.10.2 no server Server**[29]** 10.10.10.1.

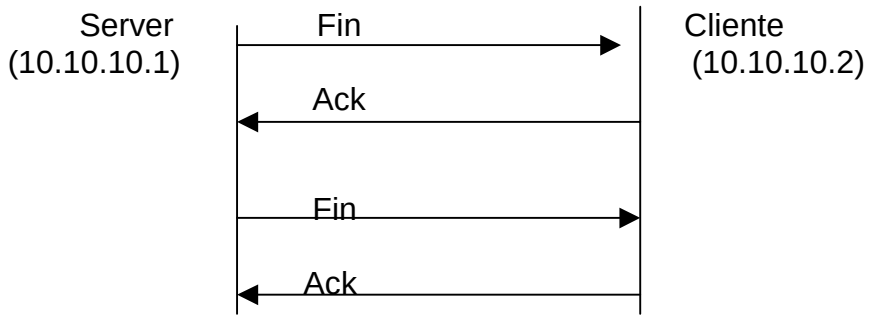
14:28:43.023855 10.10.10.1.21 > 10.10.10.2.1048: **F** 15:15(0) ack 6 win 17520 (DF) [tos 0x10]

14:28:43.025614 10.10.10.2.1048 > 10.10.10.1.21: . **ack** 16 win 8635 (DF)

14:28:43.041429 10.10.10.2.1048 > 10.10.10.1.21: **F** 6:6(0) ack 16 win 8635 (DF)

14:28:43.041917 10.10.10.1.21 > 10.10.10.2.1048: . **ack** 7 win 17520 (DF) [tos 0x10]

Neste caso estamos mostrando a finalização de uma sessão de TCP, de forma elegante.

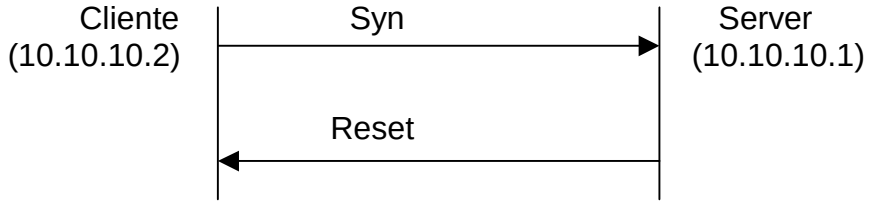


- **Finalização de maneira grosseira:** A aplicação no client 10.10.10.2 é finalizada .

16:07:22.379113 10.10.10.21138 > 10.10.10.1.23: **R** 8556518 :8556518 (0) win 0 (DF)

Neste caso o cliente**[3.2]** *telnet***[35.1]** está conectado no server; inesperadamente a "janela"deste cliente**[3.2]** é fechada sem terminar a conexão. Neste cenário o cliente**[3.2]** envia um Reset (R) avisando que a conexão foi fechada de forma agressiva independente da confirmação ou não do server .

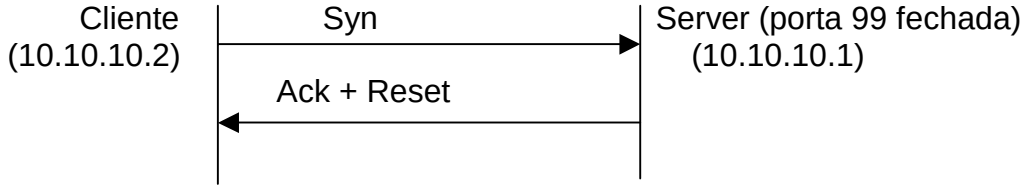
Neste caso estamos mostrando a finalização de uma sessão de TCP, de forma grosseira.



- **Porta[23] TCP fechada:** Porta**[23]** 99/TCP fecha no server**[29]** 10.10.10.1

```
15:36:24.587652 10.10.10.2.1118 > 10.10.10.1.99: S 6725117:6725117(0) win
8192 <mss 1460,nop,nop,sackOK> (DF)

15:36:24.587966 10.10.10.1.99 > 10.10.10.2.1118: R 0:0(0) ack 6725118 win 0
```



- **Porta[23] UDP fechada:** Tentativa de conexão na porta**[23]** 34/UDP .

```
15:47:27.819595 10.10.10.2 > 10.10.10.1: icmp: 10.10.10.2 udp port 34
unreachable
```

4. Método fingerprint

O método fingerprint**[7.3]** tenta identificar o sistema operacional por características da pilha TCP/IP**[34]**. São realizados alguns testes pré-determinados, e em seguida os resultados são coletados e comparados a um padrão.

Existem diversos softwares que usam essa técnica, entre eles podemos destacar: nmap**[18.5]**, queso**[24.2]**, cheops**[3.1]**, checkos (software semelhante ao nmap**[18.5]**) e etc ...

É interessante não confundir essa técnica com a simples coleta do *banner* em alguns serviços. Em alguns casos os *scans* de porta**[23]** apenas coletam banners ou saídas de comandos em serviços de telnet**[35.1]**, ssh**[31]**, http**[11]**, ftp**[9]** e outros .

Os softwares que usam a técnica fingerprint**[7.3]** comparam a resposta dos teste com uma biblioteca conhecida que deve ser atualizada com o tempo evitando que fique obsoleta.

Essa técnica é interessante para administração quando usada em conjunto com as ferramentas de mapeamento de rede, como é caso do cheops**[3.1]** .

A técnica fingerprint**[7.3]** exige um razoável conhecimento sobre protocolos**[24]** (TCP/IP**[34]**) que já foi visto no capítulo 2 e 3.

Listaremos abaixo alguns testes utilizados:

- **FIN probe:**
  É enviado um TCP**[33]** com o FIN acionado em direção a uma porta**[23]** aberta, espera –se que não haja resposta mais em alguns casos é enviado uma resposta à esta solicitação. Normalmente um TCP (Reset) .

- **BOGUS flag:**
  É enviado um pacote anômalo com flags**[8]** Syn configurado, após isso espera–se a resposta. Alguns sistemas operacionais enviam uma resposta com o Syn configurado ou um Reset .

- **TCP ISN Sampling:**
  Envia–se um pacote TCP**[33]** com um pedido de conexão e com um determinado valor no campo seq=”X”, observa–se então a resposta . Observa–se os valores de retorno e tenta classificá–los em grupos conhecidos.

- **Don't Fragment bit (DF):**
  Alguns sistemas operacionais configuram o bit**[2]** de não fragmentação, este bit**[2]** em alguns casos são configurados em momentos pré-determinados. Ao decorrer do *scan***[28]** podemos observar a resposta e verificar em qual momento o bit**[2]** é acionado.

- **Type of Service:**
  Observa–se o valor do campo TOS**[36.1]** para determinadas requisições ou envios de erros. Uma vez coletada a reposta comparamos este com um padrão conhecido.

- **TCP Options:**
  Os pacotes são enviados com os campos opções configurados, alguns sistemas operacionais não suportam todas as opções outros sim. Também

observa–se os valores de retorno e tentam enquadrá–los em um padrão conhecido.

- **TCP Initial Window:**
  Podemos determinar o sistema operacional pelo tamanho da janela (win).

- **ACK Value:**
  Observa–se o valor do bit**[2]** ACK**[1.1]** enviado, alguns sistema enviam o valor da seqüência seguindo um padrão. Os pacotes enviados podem até ter alguns flags**[8]** (Syn|Reset|Ack|Fin|Push|Urg) configurados simuntaneamente.

- **ICMP[12] Error Message Quenching:**
  Envia –se pacotes que ocasionam erros, sendo estes erros reportados por um pacote ICMP**[12]** e comparamos o envio do ICMP**[12]** de erro com algum padrão conhecido . Podemos destacar o momento o envio de UDP**[37]** para portas**[23]** fechadas. Não podemos esquecer que além deste caso os códigos do campo tipo (ICMP**[12]**) que denotam erro são: 3, 4, 5 e 12 .

- **ICMP[12] Message Quoting:**
  Vimos no capítulo 2 deste documento que o retorno de um ICMP**[12]** deve conter o cabeçalho original mais 64 Bits**[2]** do pacote original. Porém algumas implementações não seguem este padrão religiosamente.

4.1. Biblioteca do nmap[18.5]

Existe uma biblioteca que alimenta o nmap**[18.5]** para que os testes sejam efetuados, veja o arquivo: nmap-os-fingerprints**[18.5][7.3]**

O padrão de testes desta biblioteca é:

Testes	Ação
T1	Envio de um pacote Syn com TCP anômalo para portas abertas;
T2	Envio de pacote “Nulo” para portas abertas;
T3	Envio de um TCP (Syn|Fin|Urg|psh) para portas abertas;
T4	Envio de um Ack para portas abertas;
T5	Envio de um TCP (Syn) para portas fechadas;
T6	Envio de um TCP (Ack) para portas fechadas;
T7	Envio de um TCP (Fin|Psh|Urg) para portas fechadas;
PU	Envio de UDP para portas fechadas.

Dois exemplos coletados do arquivo nmap-os-fingerprints**[18.5][7.3]**:

```
Fingerprint Windows 3.1 with Trumpet Winsock 2.0 revision B
TSeq(Class=TD%gcd=10000%SI=<FF)
T1(Resp=Y%DF=N%W=1000%ACK=S++%Flags=AS%Ops=M)
T2(Resp=N)
T3(Resp=Y%DF=N%W=1000%ACK=S++%Flags=AS%Ops=M)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%
DAT=E)

Fingerprint Windows NT4 / Win95 / Win98
TSeq(Class=TD|RI%gcd=1|2|3|4|5|A|14|1E|28|5A%SI=<1F4)
T1(DF=Y%W=2017|16D0|860|869F%ACK=S++%Flags=AS%Ops=M|MNWNNT)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=2017|16D0|860|869F%ACK=S++%Flags=AS%Ops=M|MNWNNT
)
T4(DF=N%W=0%ACK=S++|O%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=S++|O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++|S%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%
DAT=E)
```

Podemos interpretar esta biblioteca para o teste do windows 3.1**[40]** como:

- Para o teste T1:

  Não fragmento não configurado;
  Janela com 1000 pacotes;
  Confirmação valores S=S+1;
  Flags**[8]** configurados no TCP são os Syn e Ack**[1.1]**;

- Para o teste T2:

  Não existe resposta;

- Para o teste T3:

  Não fragmentação não configurada;
  Janela com 1000 de tamanho;
  Confirmações S=S+1;
  Flags**[8]** do TCP configurados Syn e Ack**[1.1]**;

- Para o teste T4:

Não fragmentação não configurada;
Janela igual a zero;
Confirmações ACK=0;
Flags**[8]** configurados Reset;

- Para o teste T5:

  Não fragmentação não configurada;
  Janela igual a zero;
  Confirmações ACK S=S+1;
  Flags**[8]** configurados Ack**[1.1]** e Reset;

- Para o teste T6:

  Não fragmentação não configurada;
  Janela igual a zero;
  Confirmações ACK=0;
  Flag**[8]** configurado Reset;

- Para o teste T7:

  Não fragmentação não configurada;
  Janela igual a zero;
  Confirmações ACK**[1.1]** com mesma seqüência;
  Flags**[8]** configurados Ack**[1.1]** e Reset;

Aprendendo a escrever a biblioteca podemos editar qualquer nova descoberta concatenando –a ao arquivo de biblioteca do nmap**[18.5]**, outros scans**[28]** finger print**[7.3]** usam padrões parecidos para escrever sua biblioteca.

Um estudo apurado do nmap**[18.5]** pode ser feito ativando a opção "-O" de finger print e simultaneamente usar um sniffer**[30]** como o tcpdump**[35]** com as opções "-x –n", devemos estar preparados para um grande número de pacotes enviados e recebidos.

Esses programas que usam o método fingerprint**[7.3]** normalmente quando não obtém uma identificação positiva imprimem o resultado dos testes obtidos, com estes resultado em mãos podemos acrescentá–lo no arquivo de biblioteca referenciando–o ao sistema operacional usado no teste, que até então era desconhecido.

5. Conclusão

Esse documento tem como finalidade apresentar a teoria de protocolos**[24]** com um método simples de análise em rede. Qualquer um que desejar estudar protocolos**[24]** poderá usar um mini laboratório com um computador usando linux**[17]** ou unix**[38]** para depurar a teoria lida.

Além disso este documento simples é a base para futuros estudos sobre IDS**[13]** e análise de pacotes nas áreas de redes e/ou telecomunicações .

Com domínio nesta metodologia poderemos em seguida criar ferramentas para analisar eventos ocorridos na rede e sistemas com o intuito de fortalecer a administração do sistema e/ou aumentar a segurança.

Com o entendimento da teoria e manuseio destas ferramentas esperamos que muitos administradores passem a verificar os logs e coletas via sniffer**[30]** .

6. Bibliografia

- **Livros:**

- Computer Networks – Andrew TanenBaum – Ed. Prentice Hall PTR ;
- Redes de Computadores – Soares (PUC-Rio) – Ed. Campus ;
- TCP/IP Guia de Consulta Rápida – Luciano Palma e Rubens Prates;
- Interligação em Redes com TCP/IP (Vol. 1 e 2) – Douglas E. Comer – Ed. Campus;
- TCP/IP Tecnologia e implementação – Luís Antonio Pinto da Silva – Ed. Érica Ltda;
- TCP/IP Internet Protocolos & Tecnologias – Fernando Albuquerque – Ed. Axcel Books;
- Segurança e Prevenção em redes – Stephen Northcutt – Ed. Berkeley .

- **Sites:**

- http://www.securitylinux.com.br
- http://www.olinux.com.br
- http://www.linux.org
- http://www.freebsd.org
- http://www.openbsd.org
- http://www.netbsd.org
- http://www.conectiva.com.br
- http://www.netabc.com.br/apliavlinux/
- http://gus-br.linuxmag.com.br/
- http://www.ecst.csuchico.edu/~beej/guide/net/
- http://www.cs.vu.nl/~ast/
- http://www.gnu.org
- http://networking.earthweb.com/netsysm/article/0,,12089_920391,00.html
- http://networking.earthweb.com/netsysm/article/0,,12089_922351,00.html
- http://networking.earthweb.com/netsysm/article/0,,12089_926431,00.html
- http://tol.pro.br/
- http://www.rfc-editor.org/ (Consulta em diversas RFCs)
- http://www.nmap.org
- http://www.apostols.org/projectz/queso/
- http://packetstorm.decepticons.org/
- http://www.ussrback.com/
- http://www.securiteam.com/
- http://www.netabc.com.br/apliavlinux/
- http://www.ecst.csuchico.edu/~beej/guide/net/
- http://apliavlinux.cesio.k8.com.br/

- http://www.kimera.com.br/
- http://packetderm.cotse.com/
- http://www.securitynetworks.com.br/

7. Glossário

N°	Palavra	significado
[1].	**ARP**	Protocolo da camada de enlace, responsável pela resolução de endereços físicos e lógicos.
[1.1].	**ACK**	Flag de confirmação, usado também para indicar envio de dados;
[2].	**Bit**	Menor unidade da informação na representação numérica de base 2 (dois), o bit pode ser o (zero) ou 1 (um) sendo que estes valores são obtidos através de níveis de tensão;
[2.1].	**Binário**	Número representado na base 2;
[2.2].	**Byte**	Conjunto de 8 Bits;
[2.3].	**CCITT**	Comitê consultivo Internacional de Telegrafia e Telefonia/ITU;
[3].	**CheckSum**	String gerada pelo CRC;
[3.1].	**Cheops**	Softwares de gerencia/mapeamento de rede que usa o método fingerprint;
[3.2].	**Cliente**	Elemento que usufrui dos recursos da rede e servidor(es) em uma conexão;
[4].	**CRC**	Polinômio que verifica a integridade da informação após a trasmissão;
[5].	**DNS**	Serviço de resolução de nomes;
[6].	**Domain**	Serviço de resolução de nomes;
[7].	**Echo**	Serviço usado para verificar se uma entidade esta na rede, obsoleto. Ou resposta a uma solicitação de request (tipo 8 e cógigo 0) do ICMP ;
[7.1].	**Ethernet**	Tecnologia de rede outorgada pelo IEEE, usada em redes TCP/IP;
[7.2].	**Fin**	Flag usa no TCP para indicar finalização em uma conexão;
[7.3].	**FingerPrint**	Método de decoberta do sistema operacional através de características da implementação da pilha de protocolos TCP/IP;
[8].	**Flag**	Variável para marcar/designar um estado ou opção;

[9].	**FTP**	Serviço para transferência de arquivos. Porta 21 /TCP e Porta 20/TCP para FTP-Data;
[9.1].	**Hexadecimal**	Representação numérica na base 16, os números são representados através de: 0,1,2,...,8,9,A,B,...,F ;
[10].	**Host**	Elemento na rede usado pelo usuário, como: Computador, estação de trabalho e etc ... ;
[11].	**HTTP**	Serviço para transferência de Home Pages;
[12].	**ICMP**	Protocolo de controle de erro;
[13].	**IDS**	Sistema de detecção de invasão;
[14].	**IEEE**	Institute of Eletrical and Eletronic Engineers;
[15].	**IP**	Protocolo da camada de rede, responsável pelo endereçamento em uma conexão. Também pode ser o endereço usado pelo TCP/IP, formado por 4 octetos (4 de 8 bytes);
[16].	**ISO/OSI**	International Standards Organization;
[16.1].	**ITU**	União internacional de telecomunicações;
[17].	**Linux**	Sistema Operacional OpenSource criado por Linus Trovald (http://www.kernel.org);
[18].	**MAC**	Endereço físico de uma interface de rede;
[18.1].	**man**	Manual de comandos ou programas existentes no sistema operacional Unix ou Linux, usado através da sintaxe: man nome_do_comando;
[18.2].	**manpage**	Vide man;
[18.3].	**Mb**	MegaBits/Seg;
[18.4].	**Nibble**	Conjunto de 4 Bits;
[18.5].	**Nmap**	Scan de portas, também usa o método de fingerprint na detecção do sistema operacional;
[19].	**NTP**	Serviço para transmissão de hora apartir de uma referência. Porta 123/UDP;
[20].	**Octeto**	Grupo de 8 Bits ;
[21].	**Ping**	Ferramenta para envio de ICMP tipo 8;
[22].	**POP3**	Serviço para verificação e coleta de e-mails recebidos e armazenados no servidor de e-mail. Porta 110/TCP;
[23].	**Porta**	Endereço designado na camada de transporte após a

		multiplexação/demultiplexação dos pacotes na camada inferior, o número da porta é referenciado a um processo ou serviço do sistema operacional ou TCP/IP respectivamente. O conjunto de dados da camada de transporte e seu cabeçalho é chamado de segmento;
[24].	**Protocolo**	Linguagem para comunicação entre duas ou mais entidades. São regras definidas para a comunicação entre duas ou mais entidades;
[24.1].	**Push**	Flag usado para indicar que os dados devem ser lido rapidamente;
[24.2].	**Queso**	Ferramenta semelhante ao nmap;
[24.2].	**Reset**	Flag TCP usado para finalizar uma conexão terminada de forma repentina;
[25].	**RFC**	Recomendações sobre protocolos e serviços usados na internet;
[25.1].	**RJ-45**	Conector usado em redes com cabo Par Trançado, possui 8 Vias e pode ser blindado ou não blindado;
[26].	**ROM**	Memória somente para leitura;
[27].	**Scan**	Programa que verifica a possibilidade de conexão ou não a um host ou porta;
[28].	**Scan Port**	Software para verificar se um host ou porta(s) estão ativos, este software pode usar diversos protocolos inclusive apenas verificar se o elemento de rede esta ativo;
[29].	**Servidor**	Computador que centraliza um serviço e deve servir aos clientes da rede;
[30].	**Sniffer**	Categoria de software que capturam os dados que trafegam em um domínio de colisão ;
[31].	**SSH**	Serviço de acesso remoto com um canal encriptado. Porta 22/TCP;
[32].	**SunRPC**	Serviço criado pela SUN usado para executar procedures em um computador remoto. Porta 111/UDP;
[32.1].	**Syn**	Flag usado para indicar pedido de sincronia;
[33].	**TCP**	Protocolo da camada de transporte;
[34].	**TCP/IP**	Protocolo usado na internet, este agrupa diversos protocolos ;
[35].	**TCPDump**	Software sniffer;

[35.1].	**Telnet**	Serviço de emulação de terminal remoto, porta 23/TCP;
[36].	**TFTP**	Serviço simples de transferência de arquivos usados normalmente em equipamentos;
[36.1].	**TOS**	Tipo de serviço;
[36.2].	**Traceroute**	Ferramenta que implementa o envio do ICMP com código tipo 11 setando em x++ o TTL (camada de rede) após o retorno do mesmo quando um elemento do nível 3 (ISO/OSI) decrementa-o em x- - e este obtém valor 0 (zero);
[36.3].	**Tracert**	Mesmo que traceroute so que usado no sistema operacional windows da microsoft;
[37].	**UDP**	Protocolo da camada de transporte;
[38].	**Unix**	Sistema operacional criado no BellLab da AT&T na década de 70;
[38.1].	**URG**	Flag usado para indicar dados urgentes e que o campo “ponteiro urgente” na camada de rede deve ser lido.
[39].	**Whois**	Serviço que possibilita obter informações sobre uma rede,a consulta é solicitada aos orgãos que recebem os registros de nomes;
[40].	**Windows**	Sistema operacional da microsoft;
[41].	**X25**	Conjunto de regras definidas pela CCITT, protocolo usado em WANs para compartilhar conexão entre equipamentos.