Sequestro de Conexões TCP - Uma Abordagem Contemporânea

Felipe G. Santos*
Universidade Federal do Piauí - Campus Helvideo Nunes de Barros
Picos, Piauí
fgs4ntos@gmail.com

Rayner Gomes†
Universidade Federal do Piauí - Campus Helvideo Nunes de Barros
Picos - Piauí
rayner@ufpi.br

RESUMO

O protocolo de rede TCP, parte essencial da Arquitetura TCP/IP usada pela Internet, por décadas é estudado e testado em relação a segurança. Desde 1985 a comunidade científica vem execrando este protocolo e mostrando a facilidade de se quebrar requisitos mínimos de segurança dele. Em 1995, Paul Watson apresenta a comunidade científica a facilidade de se quebrar uma conexão TCP deixando ainda mais evidente que os 10 anos não foram suficientes para melhorar a segurança no protocolo TCP. Em 2005 Christoph Wegener e Wilhelm Dolle apresentam um novo artigo mostrando, novamente, que mais 10 anos não foram suficientes. Este trabalho refaz alguns dos testes feitos por Paul Watson orientado por uma metodologia próxima ao deles porém com Sistemas Operacionais recentes para descobrir se ainda o problema do sequestro TCP se apresenta como uma vulnerabilidade atual apesar de um quarto de século passado deste o início dos testes de segurança que o TCP vem sofrendo no tocante ao tipo de ataque de Sequestro de Conexões.

Palavras Chaves

Redes de Computadores, Ataque TCP/IP, Segurança.

1. INTRODUÇÃO

Com a infinidade de aplicações heterogenias que utilizam a Internet como meio de comunicação tais como: navegadores de página, aplicações VOIP (Voice over Internet Protocol), jogos online, e-mails e diversos outros tipos de aplicações, todas elas utilizam o protocolo TCP (Transmission Control Protocol – Protocolo de Controle de Transmissão) e portanto necessitam de uma comunicação confiável e orientada a conexão. A utilização do TCP é necessária, pois é fundamental o estabelecimento de acordos antes da transmissão de fato da mensagem (C. W. e Wilhelm Dolle [3]).

*Aluno de iniciação científica
†prof. Msc.e Orientador do projeto de iniciação científica

As conexões TCP são um conjunto de estruturas de dados que determinam os processos envolvidos numa comunicação e seus comportamentos. Estas podem ser vistas como dutos virtuais presentes nos enlaces de comunicação que filtram os milhares de pacotes que percorrem uma rede (RFC 793 e RFC 1323 [2, 1]). As aplicações que utilizam de conexões TCP precisam que estas sejam estabelecidas desde o inicio até o fim da transmissão, um provável encerramento indesejoso de uma conexão pode gerar uma situação de instabilidade e prejuízos imensuráveis.

Uma vez que a quebra de uma conexão é uma situação indesejosa espera-se que os projetistas da Arquitetura TCP reforçassem o protocolo TCP para que fosse imune a infiltrações de uma conexão ou a quebra de uma conexão por um terceiro. Mas que o mundo científico ficou abismado com a facilidade do rompimento de uma conexão. Paul Watson em seu trabalho Slipping in Windows - TCP Reset Attacks mostrou as facilidades de se quebrar uma conexão (P. Watson e Frsirt [6, 5]).

Christoph Wegenere e Wilherm Dolle [4] produziram um artigo que foi publicado na Revista Linux Magazine, outubro de 2005, que deixa bem claro que o grande problema da quebra de uma conexão está na implementação da pilha de comunicação TCP/IP feita pelos desenvolvedores de sistemas operacionais. Neste trabalho utilizou-se dois sistemas operacionais: Windows XP, Linux Kernel 2.4 e 2.6, Windows 2000 e OpenBSD 3.6, todos estes sistemas operacionais atuais para a data de 2005 quando o artigo foi publicado.

Contudo passaram-se 5 anos e novas versões do Linux, FreeBSD e Windows surgiram, muitos esforços para atender às melhorais na interação Homem-máquina foram feitas para seduzir os usuários e motivar a venda dos novos Sistemas Operacionais, porém, se a segurança quanto a quebra de conexões TCP foi melhorada é o foco que este trabalho irá investigar.

2. FUNCIONAMENTO DO PROTOCOLO TCP E A VULNERABILIDADE

O protocolo TCP teve suas especificações definidas no documento Resquest for Comments (RFC) de número 793 [2]. Este define as especificações do protocolo TCP, contemplando a estrutura do cabeçalho e as regras procedimentais.

Dentre várias características do protocolo, neste trabalho

os principais campos analisados são: a) portas de origem e destino da conexão com valores de 16 bits; b) número sequencial, que identifica o pacote de forma única na conexão com valores de 32bits; c) o número de reconhecimento para informar à outra máquina que o pacote que ela enviou anteriormente foi recebido com sucesso; d) flags de controle da conexão tais como de sincronizar (SYN), finalizar (FIN) e reset (RST); e e) o tamanho da janela de recepção sendo o ponto crítico para a base do ataque.

Uma conexão estabelecida é unicamente identificada por uma quádrupla que consiste do endereço IP e porta de origem mais o endereço IP e porta de destino. Cada pacote possui um número que o identifica e diferencia dos demais que é conhecido como o número sequencial, que serve como uma identidade. Assim é notável o perigo que isso acarreta, qualquer determinação dos detalhes da quádrupla mais o número sequencial válido tem tudo o que é preciso para entrar na conexão não importado a localização do agressor na rede.

2.1 Sequestro de Conexão TCP

É bastante simples derrubar uma conexão TCP bombardeando-a com pacotes TCP/IP, as informações necessárias para isso podem ser facilmente obtidas com o mais modesto dos sniffers: os endereços IP de origem e destino e um número sequencial TCP válido. Truques como a previsibilidade da janela, escolha de um número sequencial válido dentre os que o receptor considera válido, tornam as coisas ainda mais fáceis, a Figura 1 mostra a execução do comando netstat no Ubuntu Linux. Por meio deste comando a quádrupla da conexão é mostrada, estas informações estão presentes nos segmentos que são transmitidos na rede e facilmente capturados, basta para isto que o invasor tenha acesso ao canal pelo qual os segmentos viajam pela rede.

```
root@rayner-desktop:/home/rayner# netstat -nt
Conexões Internet Ativas (sem os servidores)
Proto Recv-Q Send-Q Endereço Local         Endereço Remoto        Estado
tcp        0      0 10.180.2.177:53865     64.233.163.104:80      ESTABELECIDA
tcp        0      0 10.180.2.177:53866     64.233.163.104:80      ESTABELECIDA
tcp        0      0 10.180.2.177:53863     64.233.163.104:80      ESTABELECIDA
tcp        0      0 10.180.2.177:53864     64.233.163.104:80      ESTABELECIDA
```

Figure 1: O comando netstat -nt.

Se o agressor conseguir obter as informações apresentada na Figura 1 uma parte do ataque está configurado. Outra parte é a descoberta no número sequencial, sendo este número de 32 bits há uma pequena chance para um acerto escolhendo um número aleatoriamente, porém, e infelizmente, algumas facilidades estão disponíveis ao agressor, um dos sistemas operacionais que escolhe de forma aleatória é o OpenBSD como mostrado na Figura 2.

2.2 O Problema do Protocolo TCP

O número sequencial pode receber um valor gigantesco, então deveria ser bem complicado para um agressor inferir um número válido. O grande problema da implementação da pilha TCP em diversos Sistemas Operacionais mostrado por Christoph Wegenere e Wilherm Dolle [4] é que este número não é escolhido aleatoriamente.

```
Active Internet connections
Proto   Recv-Q Send-Q  Local Address          Foreign Address
tcp         0      0  projetotcp.4285        dns1.ufpi.br.domain
tcp         0      0  projetotcp.38779       dns1.ufpi.br.domain
tcp         0      0  projetotcp.17096       dns1.ufpi.br.domain
Active UNIX domain sockets
```

Figure 2: O comando netstat -nt no OpenBSD 4.7.

Alem disto, o sistema de janelas de conexão apresenta-se como outro problema. A janela, que é uma margem de valores válidos, permite que o computador que está recebendo os pacotes possa colocar todos eles na ordem correta. Se o número sequencial do pacote enviado pelo invasor estiver dentro da janela de recepção, o subsistema TCP do sistema operacional irá aceitar e processar esse pacote diminuindo as possíveis tentavas de ataque para o agressor. Christoph Wegenere e Wilherm Dolle [4] descobriram que o único sistema operacional que escolhe de forma aleatória o número sequencial é o OpenBSD como mostrado na Figura 2.

Por fim, para piorar o cenário de vulnerabilidade do TCP, muitas implementações não seguem a especificação do protocolo que diz que uma conexão somente deve ser encerrada se o segmento utilizado para sinalizar o enceramento através da flag RST ou FIN estiver com o número sequencial e reconhecimento adequado.

Na próxima seção, é apresentado os testes baseados nos de Christoph Wegenere e Wilherm Dolle [4] utilizando versões recentes dos mesmos sistemas operacionais utilizados por eles, visando obter novos dados para diagnosticar se os pontos de fraqueza das implementações do protocolo TCP tais como a seleção aleatória do número sequencial inicial é de fato realizado; e se as conexões são somente encerradas segundo as regras ditadas na especificação do TCP.

3. INFRAESTRUTURA PARA O ATAQUE

Na seção 2 é apresentado os pontos de fraqueza do TCP que serão tratados neste trabalho. Evidências nos trabalhos sobre vulnerabilidade do TCP é que na grande maioria dos casos de ataque o problema está na implementação do protocolo. O não respeito das orientações e regras da especificação acaba resultando em implementações falhas, estas falhas muito bem conjugadas podem levar aos erros desejados pelos invasores. O Sequestro de Conexões consiste em derrubar uma conexão através do monitoramento dos canais de comunicação afim de obter os dados da quádrupla da conexão e inferir um número aleatório para o campo do número sequencial e do reconhecimento presentes no cabeçalho do segmento TCP. O problema é desenhado quando um agente da conexão não respeita a regra de verificar se o número utilizado pelo campo do número sequencial e número de reconhecimentos são válidos para aquele momento da conexão.

3.1 Preparação Técnica do Ambiente

O ataque é feito através do programa produto do código fonte reset-tcp.c disponibilizado por Paul Watson [6] sendo este o mesmo também utilizado por ele em seus testes. A

compilação no Ubuntu exigiu algumas instalações de programas, bibliotecas e adaptações do código. Para configuração do ambiente de execução dos ataques utiliza-se o sistema operacional Ubuntu 10.10 com o kernel 2.6.35 que exigiu a instalação de algumas bibliotecas básicas da linguagem C. A instalação destas bibliotecas foi realizada através do comando em modo de superusuário: apt-get install build-essential.

A instalação da biblioteca libnet1-dev necessária para a execução do arquivo reset-tcp.c e precisou da configuração do arquivo sources.list que se encontra no diretório /etc/apt/. As seguintes linhas foram incrementadas no final do arquivo:

- apt-get install build-essential;

Um vez configurado o arquivo sources.list a sequencia de comandos no terminal é realizado na sequência:

- deb ftp://ftp.debian.org/debian/ lenny main contrib non-free;
- deb http://debian-multimedia.org/ lenny main;

- apt-get update;
- apt-get install libnet1-dev;

3.2 Adaptação do Código reset-tcp.c

O código fonte está disponível no site da Packetstorm Security [1]. Após o download do código fonte houve uma tentativa de gerar o executável. Nesta tentativa o seguinte erro foi gerado como mostra a Figura 3.

Estes avisos gerados na saída da compilação do programa mostram que o tipo de variável usado no código somente é suportado pelo C90 (padrão ANSI C). Para não ocorrência dos mesmos basta ao final da constante 'seqmax' adicionar o sufixo UL, como apresentado na Figura 4.

```
30 int main(int argc, char *argv[])
31 {
32     int c;
33     unsigned long int count=0;
34     unsigned long int count2=0;
35     unsigned long int seqguess=0;
36     unsigned long int seqstart=0;
37     unsigned long int seqincrement=0;
38     unsigned long int seqmax=4294967295UL;
39     u_char *cp;
40     libnet_t *l;
41     libnet_ptag_t t;
42     char *payload;
43     char * device = argv[1];
```

Figure 3: Adaptação do reset-tcp na linha 38.

[1]http://packetstormsecurity.org/0404-exploits/reset-tcp.c

```
root@atacante:/home/fgs4ntos/Área de Trabalho# gcc reset-tcp.c
reset-tcp.c: In function 'main':
reset-tcp.c:38: warning: this decimal constant is unsigned only in ISO C90
reset-tcp.c:71: warning: this decimal constant is unsigned only in ISO C90
reset-tcp.c:88: warning: format '%i' expects type 'int', but argument 2 has type 'long unsigned i
nt'
/tmp/cc7bnape.o: In function `main':
reset-tcp.c:(.text+0x224): undefined reference to `libnet_name2addr4'
reset-tcp.c:(.text+0x247): undefined reference to `libnet_name2addr4'
reset-tcp.c:(.text+0x2c8): undefined reference to `libnet_init'
reset-tcp.c:(.text+0x343): undefined reference to `libnet_build_tcp'
reset-tcp.c:(.text+0x3be): undefined reference to `libnet_build_tcp'
reset-tcp.c:(.text+0x438): undefined reference to `libnet_build_ipv4'
reset-tcp.c:(.text+0x478): undefined reference to `libnet_build_ethernet'
reset-tcp.c:(.text+0x488): undefined reference to `libnet_write'
collect2: ld returned 1 exit status
root@atacante:/home/fgs4ntos/Área de Trabalho#
```

Figure 4: Erro da compilação do reset-tcp.c.

3.3 Sistemas Operacionais e Softwares de Apoio

Os testes explicados na próxima seção foram executados em máquinas utilizando os Sistemas Operacionais:

- Ubuntu 10.10, kernel 2.6.35-22-generic-pae;
- Windows Seven Ultimate;
- OpenBSD 4.7.

Alguns testes foram feitos através da virtualização de máquinas usando o VirtualBox instalado no Ubuntu. Os aplicativos de rede para monitoramento do tráfico utilizados foram:

Programa	Descrição
Ping	Programa para enviar pacotes ICMP
Netstat	Utilizado visualizado o estado das conexões.
Tcpdump	Usado para monitorar os dados dos pacotes de rede.
Wireshark	Um poderoso analisador gráfico de rede multiplataforma com suporte a diversos protocolos de rede.

4. SIMULAÇÕES E TESTES

4.1 Primeiro Teste

O primeiro teste preocupa-se em refazer o teste de Christoph Wegenere e Wilherm Dolle [4]. O ambiente de teste neste exemplo é mostrado na Figura 5. Há duas máquinas, o Servidor e o Cliente, cujos números IPs são respectivamente 192.168.1.104 e 192.168.1.102. Estas máquinas estão ligadas através de um cabo Ethernet de 1Gbps e estão conectadas usando uma topologia ponto-a-ponto. O sistema operacional utilizado é o Ubuntu 10.10 e o kernel 2.6.35-22-generic-pae.

Figure 5: Ambiente Testes 1.

Neste exemplo, é criado uma conexão com o ssh para ser alvo do ataque. O ataque é executado usando o comando

reset-tcp (o comando reset-tcp foi explicado na seção 3.1). O comando reset-tcp gera vários segmentos TCP alterando apenas o valor do número sequencial entre eles. Todos os parâmetros de funcionamento de reset-tcp são passados com valores já conhecidos e obtidos através do programa Wireshark (Figura 6), simulando desta maneira um agressor que tenha acesso ao enlace da rede. Segundo os resultados de Christoph Wegenere e Wilherm Dolle [4] a conexão, neste cenário, será encerrada.

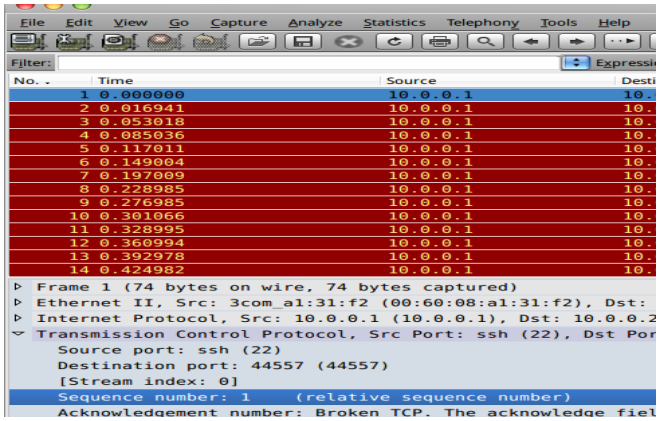

Figure 6: Obtendo dados através do Wireshark.

Logo após a conexão ssh (*shell* remoto com criptografia) entre o cliente e servidor é executado o programa reset-tcp.c (que neste teste está com o nome a.out) no servidor. A Figura 6 mostra a execução deste comando no terminal da máquina atacante. Como as máquinas estão ligadas numa rede ponto a ponto o reset-tcp poderia ser executado em qualquer uma das duas.

```
fgs4ntos@atacante:~/Área de Trabalho$ sudo ./a.out eth0 192.168.1.104 192.168.1.102 62354 1
Usage: ./a.out [interface] [src ip] [src port] [dst ip] [dst port] [window size]
**Be sure to re-compile with appropriate MAC addresses!!! You were warned.
fgs4ntos@atacante:~/Área de Trabalho$ sudo ./a.out eth0 192.168.1.104 22 192.168.1.102 62354 1
Packets sent: 8192     Sequence guess: 8191
Packets sent: 16384    Sequence guess: 16383
Packets sent: 24576    Sequence guess: 24575
Packets sent: 32768    Sequence guess: 32767
```

Figure 7: Execução do arquivo reset-tcp.c Teste 1.

4.1.1 Metodologia do Primeiro Teste

Os itens a seguir descrevem as estapas para realizar o Primeiro Teste, são eles:

1. Iniciar uma conexão ssh a partir da máquina Servidor no servidor ssh da máquina Cliente;
2. Iniciar o programa Wireshark para descobrir o número de sequencia iniciado na conexão;
3. Executar o comando reset-tcp;
4. Monitorar a conexão até ela ser derrubada.

4.1.2 Resultados Primeiro Teste

Após o estabelecimento da conexão ssh é descoberto o número de sequencial da conexão. Desta forma houve uma grande expectativa do breve encerramento da conexão pois ele foi um número baixo valor, uma vez que o programa reset-tcp inicia sua sequencia de repetições a partir do 0.

Logo após o início da execução do programa a conexão não foi encerrada contrariando a expectativa. Monitorando o canal de comunicação com o WireShark há a certeza que os pacotes estão sendo ejetados na rede e que os valores corretos para derrubamento estão sendo usados.

Depois de alguns minutos, durante este tempo mais de 32.000 pacotes foram inseridos na rede sem sucesso, outras tentativas foram feitas. Em todas as tentativas o número sequencial inicial foi baixo indicando que se o programa funcionasse como esperado os testes seriam rápidos, mas em todos os testes o programa foi encerrado sem sucesso.

4.2 Segundo Teste

O Segundo teste tem o mesmo objetivo do Primeiro Teste (Seção 4.1), ou seja, derrubar uma conexão estabelecida. A diferença neste teste é o Sistema Operacinal de uma das máquinas. No primeiro teste a máquina Cliente esta com o Linux e neste teste o sistema operacional é substituído por Linux.

O ambiente de teste neste exemplo é mostrado na Figura 8. Há duas máquinas, a máquina nomeada de Servidor e a outra nomeada de Cliente cujos números IPs são respectivamente 192.168.1.104 e 192.168.1.107. A máquina Cliente é uma máquina virtual, seu sistema operacional é o *Windows 7 Ultimate*. A máquina Servidor é a atacante, assim como no primeiro teste, o sistema operacional utilizado é o Ubuntu 10.10 e o kernel 2.6.35-22-generic-pae. A topologia utilizada neste teste também é a ponto-a-ponto.

Figure 8: Ambiente Testes 2.

Logo após o estabelecimento de conexão ssh entre as máquinas Cliente e Servidor (Figura 8) é executado o arquivo reset-tcp.c. O ataque procede da máquina Servidor com Linux para a máquina Cliente que usa o Windows (Figura 9).

4.2.1 Metodologia do Segundo Teste

A metodologia usada no segundo teste foi a mesma utilizada no Primeiro Teste (Seção 4.1):

1. Iniciar uma conexão ssh a partir da máquina Servidor no servidor ssh da máquina Cliente;
2. Iniciar o programa Wireshark para descobrir o número de sequencia iniciado na conexão;

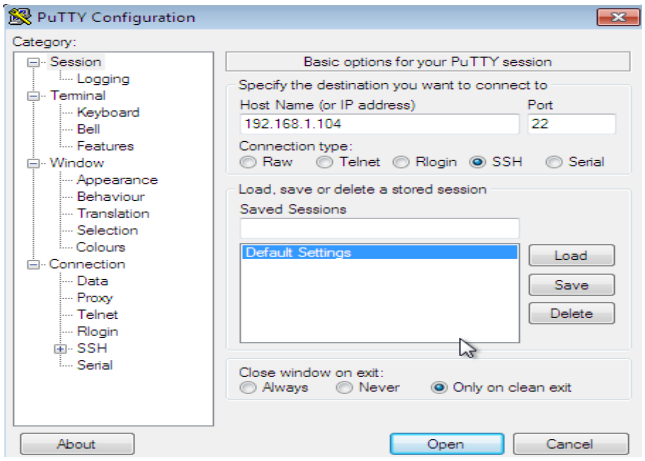

Figure 9: Tela do Putty.

```
fgs4ntos@atacante:~/Área de Trabalho$ sudo ./a.out eth0 192.168.1.104 192.168.1.107 49158 1
Usage: ./a.out [interface] [src ip] [src port] [dst ip] [dst port] [window size]
**Be sure to re-compile with appropriate MAC addresses!!! You were warned.
fgs4ntos@atacante:~/Área de Trabalho$ sudo time ./a.out eth0 192.168.1.104 22 192.168.1.107 49158
1
Packets sent: 8192     Sequence guess: 8191
Packets sent: 16384    Sequence guess: 16383
Packets sent: 24576    Sequence guess: 24575
Packets sent: 32768    Sequence guess: 32767
```

Figure 10: Execução do arquivo reset-tcp.c Teste 2.

3. Executar o comando reset-tcp;
4. Monitorar a conexão até ela ser derrubada.

4.3 Resultados Segundo Teste

Na utilização do Windows 7 para realização dos testes de sequestro de conexão foi obtido os mesmos resultados encontrados no teste realizado no sistema operacional Ubuntu 10.10, ou seja, mesmo inferindo os valores corretos para a quádrupla IP e porta de origem mais IP e porta de destino mais o número sequencial a conexão não foi encerrada.

4.4 Terceiro Teste

O terceiro teste é muito parecido com o Segundo Teste (Seção 4.2), apenas é invertido a posição do servidor ssh entre a máquina com Windows e Linux. Assim, o servidor é o computador com o Windows e o cliente é a máquinas com o Linux.

4.4.1 Metodologia do Terceiro Teste

A metodologia usada no terceiro teste foi a mesma utilizada no Segundo Teste (Seção 4.2):

1. Iniciar uma conexão ssh a partir da máquina Servidor no servidor ssh da máquina Cliente;
2. Iniciar o programa Wireshark para descobrir o número de sequencia iniciado na conexão;
3. Executar o comando reset-tcp;
4. Monitorar a conexão até ela ser derrubada.

4.4.2 Resultados Terceiro Teste

Na utilização do Windows 7 como Servidor e o Linux como Cliente para realização do teste de sequestro de conexão foi obtido o mesmo resultado encontrado no teste da seção anterior (Seção 4.2). Mesmo inferindo os valores corretos para a quádrupla IP e porta de origem mais IP e porta de destino mais o número sequencial a conexão não foi encerrada.

5. CONCLUSÃO

O que fica evidente é que o Ubuntu 10.10 com o kernel 2.6.35-22-generic-pae e *Windows 7 Ultimate* respeitam uma das regras de segurança do TCP (RFC 793), até então não respeitada nos teste realizados por Christoph Wegener e Wilhelm Dolle, que para derrubar uma conexão é necessário saber também o número do reconhecimento (número ack) esperado pela outra parte da comunicação.

Christoph Wegener e Wilhelm Dolle relataram em seu artigo Sequestro de Conexões TCP/IP que o Linux (testado em um Fedora Core 3 com *kernel* 2.6.9) encerrava uma conexão apenas utilizando a quádrupla IP de origem, IP de destino, porta de origem e porta de destino mais o número da sequencia esperado pelo parceiro da comunicação. Em seus testes foi provado que o número de reconhecimento (número ack) não é levado em consideração contrariando as regras de segurança da especificação do TCP.

Os sistemas testados impõem uma segurança muito maior que a versões passadas testadas por Christoph e Wilhelm, e com isto o programa Reset-TCP precisa ser reescrito para que volte a sua finalidade, da forma o qual está escrito sob as plataformas atuais, uma conexão jamais será derrubada.

Para total comparação com o trabalho de Christoph Wegener e Wilhelm Dolle ainda é necessário testar com o sistema operacional OpenBSD (versão mais recente).

Como trabalho futuro, o grupo de pesquisa visa buscar nos códigos fontes dos sistemas operacionais abertos utilizados neste trabalho trecho de códigos referente ao processo de finalização de uma conexão presente na camada de rede. Como apresentado no artigo os sistemas operacionais utilizados estão de acordo com a especificação e respeitam a cláusula de verificar o valor correto da *flag sync* e *ark* para o encerramento de uma conexão.

Por fim, seguindo objetivo deste artigo de testar a vulnerabilidade do TCP, vulnerabilidade esta conhecida como sequestro TCP, também será extendido os testes para atuais sistemas operacionais abertos utilizados em aparelhos de telefonia móvel, como por exemplo o Android.

6. AGRADECIMENTOS

Agradecemos ao programa de iniciação científica da Pró-Reitoria de Pesquisa e Pós-Graduação (PRPPG [2]) mantida pela Universidade Federal do Piauí (UFPI [3]) pelo incentivo na execução deste trabalho.

[2] www.ic.ufpi.br
[3] www.ufpi.br

7. REFERÊNCIAS

[1] R. 1323. Tcp extensions for high performance. julho 2010. http://tools.ietf.org/html/rfc1323.

[2] R. 793. Transmission control protocol. julho 2010. http://tools.ietf.org/html/rfc1323.

[3] O. S. V. Database. Slipping in the windows. tcp reset attacks. Julho 2010. http://osvdb.org/ref/04/04030-SlippingInTheWindow$_v$1.0.*doc*.

[4] C. W. e Wilhelm Dolle. Entendendo e evitando ataques ao protocolo tcp. pages 66–73, Outubro 2005. www.linuxmagazine.com.br/.../LM13$_T CPhijack.pdf$.

[5] Frsirt. Tcp connection reset remote exploit. julho 2010. www.frsirt.com/exploits/04232004/tcpexploit.

[6] P. Watson. Open source vulnerability database. julho 2010. http://osvdb.org/ref/04/04030-SlippingInTheWindow$_v$1.0.*doc*.