MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA

FÁBIO PINHÃO MELLO
RICARDO DA CRUZ MENDES JUNIOR
DANIEL GUIMARÃES ROCHA

ANÁLISE DE ATAQUES DDOS

Rio de Janeiro
2010

INSTITUTO MILITAR DE ENGENHARIA

FÁBIO PINHÃO MELLO
RICARDO DA CRUZ MENDES JUNIOR
DANIEL GUIMARÃES ROCHA

ANÁLISE DE ATAQUES DDOS

Monografia de Projeto de Fim de Curso apresentado ao Curso de Engenharia de Computação do Instituto Militar de Engenharia.

Orientador: Cap ANDERSON Fernandes P. dos Santos, D. Sc.

Rio de Janeiro
2010

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 - Praia Vermelha
Rio de Janeiro - RJ - CEP: 22290-270

Mello, Fábio Pinhão
Junior, Ricardo da Cruz Mendes
e
Rocha, Daniel Guimarães
ANÁLISE DE ATAQUES DDOS / Fábio Pinhão Mello, Ricardo da Cruz Mendes Junior e Daniel Guimarães Rocha - Rio de Janeiro: Instituto Militar de Engenharia, 2010.

Monografia de Projeto Final de Curso - Instituto Militar de Engenharia, 2010.

1. Formação.

INSTITUTO MILITAR DE ENGENHARIA

FÁBIO PINHÃO MELLO
RICARDO DA CRUZ MENDES JUNIOR
DANIEL GUIMARÃES ROCHA

ANÁLISE DE ATAQUES DDOS

Monografia de Projeto de Fim de Curso apresentado ao Curso de Engenharia de Computação do Instituto Militar de Engenharia.

Orientador: Cap Anderson Pereira dos Santos, D. Sc.

Aprovada em 16 de Abril de 2010 pela seguinte Banca Examinadora:

Cap ANDERSON Fernandes P. dos Santos - D. Sc.
Orientador

Raquel Coelho Gomes Pinto - D. Sc.

Maj Sérgio dos Santos CARDOSO Silva - M. Sc.

Rio de Janeiro
2010

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AFRL	*Air Force Research Laboratory*
API	*Application Programming Interface*
BKIS	*Bach Khoa Internet Security*
CAIS	*Centro de Atendimento a Incidentes de Segurança*
CGI	*Common Gateway Interface*
CPU	*Central Processing Unit*
DAN	*Direct Action Network*
DARPA	*Defense Advanced Research Projects Agency*
DDoS	*Distributed Denial Of Service*
DNS	*Domain Name System*
DoS	*Denial Of Service*
GUI	*Graphical User Interface*
HTTP	*Hyper Text Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IP	*Internet Protocol*
IRC	*Internet Relay Chat*
ISP	*Internet Service Provider*
JVM	*Java Virtual Machine*
KDD	*Knowledge Discovery in Databases*
MIT	*Massachusetts Institute of Technology*
OTAN	Organização do Tratado do Atlântico Norte
QoS	*Quality of Service*
SYN	*Synchronization*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*

RESUMO

Em um ataque de negação de serviço, um atacante tenta impedir usuários legítimos de acessar informações ou serviços de uma máquina servidora. Um dos tipos mais comum de ataque ocorre com a inundação de uma rede com pacotes de informações. A versão distribuída de um ataque de negação de serviço, chamada de ataque de negação de serviço distribuído (*DDoS*), que ocorre quando múltiplos sistemas realizam a inundação de uma rede da máquina servidora alvo ao mesmo tempo, tornou-se uma arma difundida nos conflitos cibernéticos contemporâneos. Este trabalho tem por objetivo fornecer um *trace* de um ataque *DDoS* e preparar uma estrutura para servir de base para testes futuros de outros ataques. Este objetivo foi alcançado através de vários tipos de experimentos de bombardeamento de pacotes em laboratório de forma a simular ataques de negação de serviço.

ABSTRACT

In a denial of service attack, an attacker tries to stop legitimate users from accessing information or services from a server machine. One of most common type of attack is the packet-flooding over a network. The distributed version of a denial of service attack, which is called distributed denial of service (*DDoS*) and occurs when multiple systems perform the flooding over a network of a target server machine at the same time, became a widespread weapon in contemporary cyberconflicts. This paper aims to provide a *DDoS* attack trace and prepare a framework to serve as a basis for future testing of other attacks. This goal was achieved by different types of packet-bombarding experiments made in the laboratory in order to simulate many denial of service attack.

1 INTRODUÇÃO

Os ataques de “negação de serviço” ou *DoS* são bastante conhecidos no âmbito da comunidade de segurança de redes. Estes ataques, através do envio indiscriminado de requisições a um computador alvo ou da exploração de uma vulnerabilidade em um protocolo, visam causar a indisponibilidade dos serviços oferecidos por ele. Fazendo uma analogia simples, é o que ocorre com as companhias de telefone nas noites de natal e ano novo, quando milhares de pessoas decidem, simultaneamente, cumprimentar à meia-noite parentes e amigos no Brasil e no exterior. Nos cinco minutos posteriores à virada do ano, simplesmente não se consegue completar a ligação, pois as linhas telefônicas estão saturadas. Além disso, outra maneira de fazer esses ataques é provocar um disfuncionamento do sistema através do envio de dados não previstos em deteminado protocolo, como por exemplo, o *ping of death* que consiste em criar um datagrama *IP* cuja tamanho total excede o tamanho máximo autorizado (65536 *bytes*). Um pacote deste tipo, enviado a um sistema mais vulnerável, faria com que este não funcionasse corretamente. [2]

Desde 1999, uma categoria de ataques de rede tem-se tornado bastante conhecida: o ataque de negação de serviço distribuído (*DDoS*). Neste novo enfoque, os ataques não são baseados no uso de um único computador para iniciar um ataque, no lugar são utilizados centenas ou até milhares de computadores ligados na *Internet* para lançar coordenadamente o ataque. A tecnologia distribuída não é completamente nova, no entanto, vem amadurecendo e se sofisticando de tal forma que até mesmo *script kiddies* (vândalos curiosos e sem muito conhecimento técnico) podem causar danos sérios. A este respeito, o Centro de Atendimento a Incidentes de Segurança (*CAIS*) tem sido testemunha do crescente desenvolvimento e uso de ferramentas de ataque distribuídas, em várias categorias: *sniffers*, *scanners*, *DoS*. [2]

Os ataques *DDoS*, nada mais são do que o resultado de se conjugar os dois conceitos: negação de serviço e intrusão distribuída. Segundo a Rede Nacional de Ensino e Pesquisa (*RNP*), os ataques *DDoS* podem ser definidos como ataques *DoS* diferentes partindo de várias origens, disparados simultânea e coordenadamente sobre um ou mais alvos. De uma maneira simples, ataques *DoS* em larga escala. [2]

Um ataque de negação de serviço é composto por um *atacante*, ou seja, aquele que deseja causar a negação de serviço. Este possui sob seu comando um ou mais computadores *mestre*, cada mestre possui sob seu comando até milhares de computadores, chamados de *escravos*. O ataque consiste em fazer com que os escravos se preparem para acessar um determinado recurso em um determinado servidor em um mesmo instante. Por exemplo, servidores *Web* possuem um número limitado de usuários que pode atender simultaneamente, o grande e repentino número de requisições

de acesso esgota esse número máximo, fazendo com que o servidor não seja capaz de atender a mais nenhum pedido. Dependendo do recurso atacado, o servidor pode chegar a reiniciar ou até mesmo ficar travado. Por exemplo, em 2009, o *Twitter* e o *Facebook* sofreram um ataque e chegaram a ficar fora do ar por um dia [3].

Para atingir uma grande quantidade de computadores conectados à *Internet*, foram criados *worms* com a intenção de disseminar pequenos programas para ataques *DDoS*. Assim, quando um vírus com tal poder contamina um computador, este fica disponível para fazer parte de um ataque *DDoS* e o usuário dificilmente fica sabendo que sua máquina está sendo utilizado para tais fins. [4]

Worms conhecidos criados para a distribuição de rotinas de ataque de negação de serviço incluem *Codered*, *Slammer*, *MyDoom*, *MyPenis*, *MyBalls* e *Trojan Horse* ou Cavalo de Tróia , que escravizam o infectado. Ferramentas conhecidas de ataques *DDoS* incluem *Fabi* (1998), *Blitznet*, *Trin00* (jun/1999), *TFN* (ago/1999), *Stacheldraht* (set/1999), *Shaft*, *TFN2K* (dez/1999), *Trank*. [4]

Os primeiros ataques *DDoS* documentados surgiram em agosto de 1999. No entanto, esta categoria se firmou como a mais nova ameaça na *Internet* na semana de 7 a 11 de fevereiro de 2000, quando vândalos cibernéticos deixaram inoperantes por algumas horas sites como o *Yahoo*, *EBay*, *Amazon* e *CNN*. Uma semana depois, teve-se notícia de ataques *DDoS* contra sites brasileiros, tais como: *UOL*, *Globo On* e *IG*, causando com isto uma certa apreensão generalizada. [2]

1.1 Histórico

Há muito se fala do uso da informação como ferramenta de ação política. Em meados do século XX o assunto foi abordado pelo arquiteto e urbanista Paul Virilio, nascido em Paris em 1932, que consagrou grande parte de seus estudos à análise do impacto social da revolução tecnológica e deu origem ao termo *bomba informática*, onde a quantidade de informações seria tão grande e vital para uma sociedade que os governos e os poderes militares a usariam como meios de guerra. Não se compreenderia nada, efetivamente, à desregulamentação sistemática da economia mundial sem estabelecer uma correlação com a desregulamentação sistêmica da informação. [5]

Tempos depois, John Arquilla e David Ronfeldt publicaram um artigo entitulado *Cyberwar is Coming!* [6]. Para Arquilla e Ronfeldt a luta pelo futuro que faz o cotidiano de nossas manchetes não está sendo travada por exércitos liderados por estados ou sendo conduzida por imensas e milionárias armas feitas para os tanques, aviões ou esquadras. Elas se desenvolvem através de grupos que operam em unidades pequenas e dispersas, podendo se desdobrar repentinamente em qualquer lugar ou tempo como uma incontrolável infecção por afluência popular. Eles sabem como dispersar,

penetrar e romper ou eludir. Os combatentes podem pertencer a redes de terroristas como a *Al Qaeda*, redes de traficantes como *Cali*, redes de militantes anarquistas como o *Black Bloc*, redes de luta política como o *Zapatismo* ou redes de ativistas da sociedade civil global como o *DAN*. Para compreender este modo emergente de luta e conflito, surgido na sociedade contemporânea a partir da revolução tecnológica que construiu a infraestrutura do ciberespaço, Arquilla e Ronfeldt criaram em 1993 - mesmo ano do surgimento do conceito de comunidade virtual - o conceito de rede de guerra (*netwar*), como o oposto correlato do conceito de ciberguerra (*cyberwar*), também por eles gerado na mesma ocasião, ambos constituindo a maior parte do campo da infoguerra (*infowar*) no mundo atual [7]. Enquanto a ciberguerra compreenderia a luta de alta intensidade conduzida através de alta tecnologia militar travada por dois estados, como, por exemplo, a Guerra do Golfo, a rede de guerra seria a luta de baixa intensidade travada de modo assimétrico por um estado e grupos organizados em rede através do uso de táticas e estratégias que envolvem o intenso uso das novas tecnologias comunicacionais da *Internet*.

Entre 1999 e 2003 o governo dos Estados Unidos sofreram uma série de ataques *DDoS* coordenados, esta série de ataques foi chamada de "Chuva de Titãs" (*Titan Rain*). Os ataques são ditos terem sido realizados por chineses, mesmo que sua verdadeira natureza (vindos de espionagem corporativa, espionagem de estado ou ataques de *hackers* amadores) e suas reais identidades (mascarados por *proxy* ou computadores *zumbi*) não fossem confirmadas. Em meados de dezembro de 2005, o diretor do *SANS Institute*, um instituto de segurança dos Estados Unidos, disse que os ataques foram provavelmente provindos de *hackers* militares chineses tentando coletar informações dos sistemas americanos [8]. *Hackers* da *Titan Rain* tiveram acesso a muitas redes de computadores dos Estados Unidos, incluindo redes da *Lockheed Martin*, *Sandia National Laboratories*, *Redstone Arsenal* e *NASA*.

Em 2007, o governo dos Estados Unidos foi vítima de um grande evento de espionagem cibernética [9], onde uma potência estrangeira desconhecida quebrou todas as agencias de alta tecnologia, as agências militares e foram baixados *terabytes* de informações. Relatos do ex-chefe de Inteligência Nacional dos Estados Unidos, Jim Lewis, foram: "*Em 2007 nós provavelmente tivemos nossa versão cibernética do ataque "Pearl Harbor". Algum poder estrangeiro desconhecido, que honestamente, não sabemos quem é, invadiu o Ministério de Defesa Americano, o Departamento de Estado, o Departamento de Comércio e, provavelmente, o Departamento de Energia, e provavelmente ainda a NASA. Eles invadiram todas as agências de alta tecnologia e todas as agências militares, transferido terabytes de informação.*"

Várias fontes de inteligência proeminente confirmaram que houve uma série de ataques cibernéticos no Brasil [9]: um ao norte do Rio de Janeiro em janeiro de 2005

que afetou três cidades e dezenas de milhares de pessoas, e outra, a partir de eventos muito maiores em 26 de setembro de 2007. Um ataque, no estado do Espírito Santo afetou mais de três milhões de pessoas em dezenas de cidades ao longo de um período de dois dias, causando grandes transtornos. Em Vitória, o maior produtor mundial de minério de ferro teve sete plantas que ficaram *offline*, custando à empresa sete milhões de dólares. Até hoje, não foram descobertos os causadores do ataque e qual era o motivo.

Em maio de 2007, uma onda de três semanas de enorme cyber-ataques contra o pequeno país do Báltico, Estónia, foi motivo de alarme em toda a aliança ocidental. A *OTAN*, na época, analisou urgentemente a ofensiva e as suas implicações. Enquanto a Rússia e a Estónia estavam envolvidos em sua pior disputa desde o colapso da União Soviética, uma linha que estourou no final do mês de abril de 2007 devido a remoção do memorial da guerra na Central de Tallinn, uma estátua de bronze de um soldado soviético, por estonianos, o país foi submetido a um ataque de negação de serviço distribuído de larga escala, derrubando sites dos ministérios, partidos políticos, jornais, bancos e empresas. A Otan enviou alguns dos seus maiores especialistas em terrorismo cibernético para Tallin para investigar e ajudar os estonianos a fortalecer suas defesas eletrônicas. As relações entre o Kremlin e o Ocidente passaram por uma crise devido a Rússia estar envolvida em disputas amargas, não só com a Estónia, mas com a Polónia, Lituânia, República Tcheca e com a Geórgia - todas as partes da antiga União Soviética ou ex-membros do o Pacto de Varsóvia.

Em meio ao conflito russo-georgiano, após semanas de especulações em torno de fóruns de Internet russos chegaram a conclusão que houve um cyber-ataque russo coordenado contra infra-estruturas de *Internet* da Geórgia em agosto de 2008. Os ataques comprometeram vários sites do governo georgiano com contínuos ataques *DDoS*, o que levou o governo a mudar seus servidores de hospedagens locais para os Estados Unidos, com o Ministério de Negócios Estrangeiros da Geórgia tomando um passo desesperado, a fim de divulgar informações em tempo real movendo-se para uma conta estrangeira.

Em julho de 2009, houve uma série coordenada de ataques de negação de serviço contra servidores do governo, imprensa e sites financeiros na Coréia do Sul e os Estados Unidos [10]. Acreditava-se que a Coréia do Norte fosse a responsável, contudo um pesquisador traçou os ataques originando-se no Reino Unido. Nguyen Minh Duc, diretor sênior de segurança da *BKIS*, disse que ganhou o controle de dois dos oito servidores atacados e por isso foi capaz de descobrir o servidor atacante mestre. Através do intervalo de endereço *IP* do servidor, identificou-se que os rastros vinham de endereços registrados no Reino Unido, a empresa não pôde ser imediatamente contatada. Pela análise dos arquivos de registros históricos dos dois servidores

que controla, o diretor disse que os ataques utilizaram cento e sessenta e seis mil computadores em setenta e quatro países que tinham sido infectados. Esse número é significativamente maior do que o número que outras empresas de segurança haviam estimado. O maior número de computadores infectados foi na Coréia do Sul seguido os Estados Unidos, China, Japão, Canadá, Austrália, Filipinas, Nova Zelândia, Reino Unido e Vietnã.

Em dezembro de 2009, um ataque cibernético, batizada *Operação Aurora*, foi lançado da China contra o *Google* e mais de 20 outras empresas [11]. O ataque atingiu uma ampla gama de empresas de vários setores - finanças, tecnologia, mídia e setores. Supõe-se que o objetivo principal dos atacantes era acessar contas do *Gmail* do *Google* e outros servidores de correio de chineses defensores dos direitos humanos. Em abril de 2010, devido a diferenças com as políticas agressivas e censurativas do governo chinês, a *Google* finalizou seus serviços na China com o desligamento da *Google.cn* e a retirada se seus escritórios no país.

1.2 Motivação

É da natureza humana a existência de discórdia entre grupos que interagem entre si. A *Internet* hoje não só permite esta interação, como jamais se permitiu na história da humanidade, como também oferece serviços que fornecem às pessoas rendimentos, educação, entretenimento e informação. A interrupção de um servidor *Web* ou de um roteador pode ser definitiva na interferência de um ou vários destes serviços. Com a popularização da rede mundial de computadores, ela tornou-se não só um local de encontro de pessoas do mundo todo, mas também um palco de conflitos. Neste contexto se encaixa o *DDoS*. [12]

Todavia, os ataques não são um fim em si. Os primeiros ataques de *DDoS* eram feitos por *hackers* apenas para provar que a segurança de um *website* nunca era o bastante para que nunca pudesse ser quebrada, e, a partir disso, travar brigas entre si pela supremacia dos atacantes, via *DDoS*. Também ocorrem ataques políticos e ataques de extorsão, cobrando os *websites* por proteção, prática que remonta às da máfia. [12]

Mas hoje não são só os *hackers* que realizam estes ataques. Dadas as facilidades de obter ferramentas na *Internet*, muitos usuários comuns tornam-se atacantes, e mesmo os próprios países, usando das máquinas do Estado, amplos recursos e inteligência, podem desenvolver ferramentas próprias, inclusive para atacar outros países em guerras. Diante destes fatos, deve-se desmistificar o ataque, de modo que administradores, gerentes de sistemas e até usuários comuns, conhecendo melhor o inimigo, se preparem para combatê-lo. [12]

Diariamente estão sendo feitos estudos para obter mais informações sobre esses ataques, porém a base de dados utilizada não é muito atual, além disso, não existe uma estrutura prévia montada para dar suporte a essas pesquisas gerando trabalho extra. Portanto, hoje se faz necessária uma base de dados atual que possa representar melhor o contexto dos ataques e uma estrutura simples que suporte diferentes tipos de testes para ataque e defesa, facilitando e acelerando o desenvolvimento do conhecimento e de produtos.

1.3 Objetivos

Esse trabalho tem como objetivo principal fornecer uma base de dados de um ataque e preparar uma estrutura simples para servir como base para testes futuros de outros ataques. Além disso, far-se-á uma experiência no laboratório enviando o maior número de pacotes possível de forma a simular um ataque.

Para atingir estes objetivos, outros objetivos secundários foram traçados. Realizar uma pesquisa sobre o atual estado da Guerra Cibernética e de Ataques de Negação de Serviços.

1.4 Estrutura do Trabalho

Esta monografia está organizada de forma a proporcionar um entendimento mais natural do assunto.

No capítulo 2 encontra-se a definição de *DDoS* bem como as classificações e taxonomia dos ataques e defesas de negação de serviço.

No capítulo 3 será exposto o ataque que será utilizado para o experimento. Será feito uma análise completa do ataque, identificando características e peculiaridades.

No capítulo 4 será mostrado o experimento realizado e apresentado os resultados no capítulo 5.

Por fim, no capítulo 6, serão feitas as considerações finais sobre o projeto e também serão apresentadas perspectivas de passos futuros do desenvolvimento do projeto.

2 ATAQUES DE NEGAÇÃO DE SERVIÇO

2.1 Descrição do que é um Ataque

Ataque é o ato de tentar dominar ou manipular um sistema. Os ataques dividem-se em duas categorias: passivos ou ativos. Ataques passivos são mais difíceis de serem identificados, geralmente são usados para monitorar um canal de comunicação, obter informações do sistema ou ler dados sem alterá-los. Já os ataques ativos são mais fáceis de se identificar, normalmente visam: modificar ou excluir dados; comprometer a disponibilidade ou a integridade de um sistema.

DDoS é um tipo de ataque onde se busca impedir que usuários verdadeiros, de um determinado sistema, utilizem um serviço ou uma rede do sistema [13]. Tem como característica principal a exploração de uma vulnerabilidade de protocolo ou o envio indiscriminado de requisições a um ou mais serviços de um ou mais servidores alvo, para que, em consequência disso, cause a indisponibilidade do referente serviço [2].

2.1.1 Os Atores

Os atores são os componentes necessários na estrutura de um ataque *DDoS*. Esses componentes são melhor visualizados na FIG. 2.1. A definição desses atores, neste trabalho, seguirá a seguinte terminologia: [2]

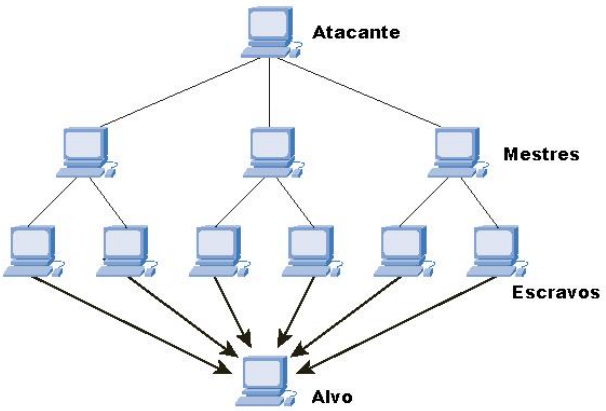


FIG. 2.1: Estrutura de um ataque *DDoS* (extraído de *www.cisco.com*)

- *Atacante:* Máquina que comanda o ataque, tem sobre seu comando os *mestres*;
- *Mestre:* Máquina que recebe ordens do *atacante* e as repassa para todos os seus *escravos*;
- *Cliente:* Aplicação residente em cada *mestre* usada para receber as ordens do *atacante* e repassá-las aos seus *escravos*, enviando as instruções para os respectivos *daemons*;

- *Escravo:* Máquina que recebe ordens de seu *mestre* correspondente e efetiva o ataque contra um ou mais alvos, conforme definido pelo *atacante*;
- *Daemon:* Aplicação residente em cada *escravo* usada para receber as ordens de seu *mestre* e executá-las, efetivando assim o ataque;
- *Alvo:* Máquina que será a vítima do ataque.

2.1.2 Fases

Para facilitar a visualização do funcionamento de um ataque *DDoS* pode-se considerar o ataque em três fases: [2]

1. *Captura de mestres e escravos:*
    - É realizado um *escaneamento* de portas e vulnerabilidades em redes que possuam alta velocidade de conexão ou baixo grau de monitoramento;
    - As vulnerabilidades encontradas são exploradas e, a partir disso, é obtido o acesso privilegiado nessas máquinas;
    - É gerada uma lista com os *IPs* de todas as máquinas capturadas.
2. *Instalação de daemons e clientes:*
    - Separam-se as máquinas que serão *mestres* e as que serão *escravos*. Em geral, as máquinas delegadas como *mestre* são as menos monitoradas, sem necessariamente possuir alta velocidade de conexão. Já os *escravos*, normalmente, são as máquinas com conexões de alta velocidade;
    - Os *daemons* e *clientes* são instalados em seus respectivos proprietários, para ser possível o seu acesso remoto;
    - Os aplicativos são executados e, logo, os *daemons* anunciam seu estado ativo aos *clientes* e permanecem em estado de espera, e então, os *clientes* registram os *IPs* das máquinas, sob seu comando, que se anunciaram.
3. *Acionamento do ataque:*
    - O *atacante* ordena o ataque aos *mestres*, que, por sua vez, ordenam aos *escravos*, e então, o ataque é disparado contra um ou mais *alvos*, conforme determinado pelo *atacante*.

2.2 Taxonomias

As taxonomias tem por objetivo classificar, tanto ataques quanto mecanismos de defesa, em grupos que possuam características distintas. Os três principais fatores que levam uma taxonomia a ser o mais abrangente possível são:

- Cada ataque, assim como, cada sistema de defesa deve ser categorizado em um único grupo;

- Para todos os possíveis tipos de ataque e todos os possíveis mecanismos de defesa deve existir uma classificação pelas respectivas taxonomias propostas;
- A taxonomia pode ser expandida no futuro, detalhando mais as ramificações de cada grupo.

Neste trabalho, serão adotadas, totalmente, as taxonomias, de ataque e de mecanismos de defesa, propostas em [14]. A razão de tal escolha se origina no fato de que as taxonomias propostas, no artigo já referenciado, são, de fato, completas nos seguintes aspectos: a taxonomia de ataques abrangem não só os ataques conhecidos como também aqueles que ainda não foram estudados, mas que, atualmente, são ameaças em potencial que poderiam afetar os mecanismos de defesa existentes; a taxonomia de sistemas de defesa abrange não somente as abordagens existentes mas também algumas de uso comercial que foram documentadas o suficiente para serem analisadas. [15]

2.2.1 Taxonomia de Ataque *DDoS*

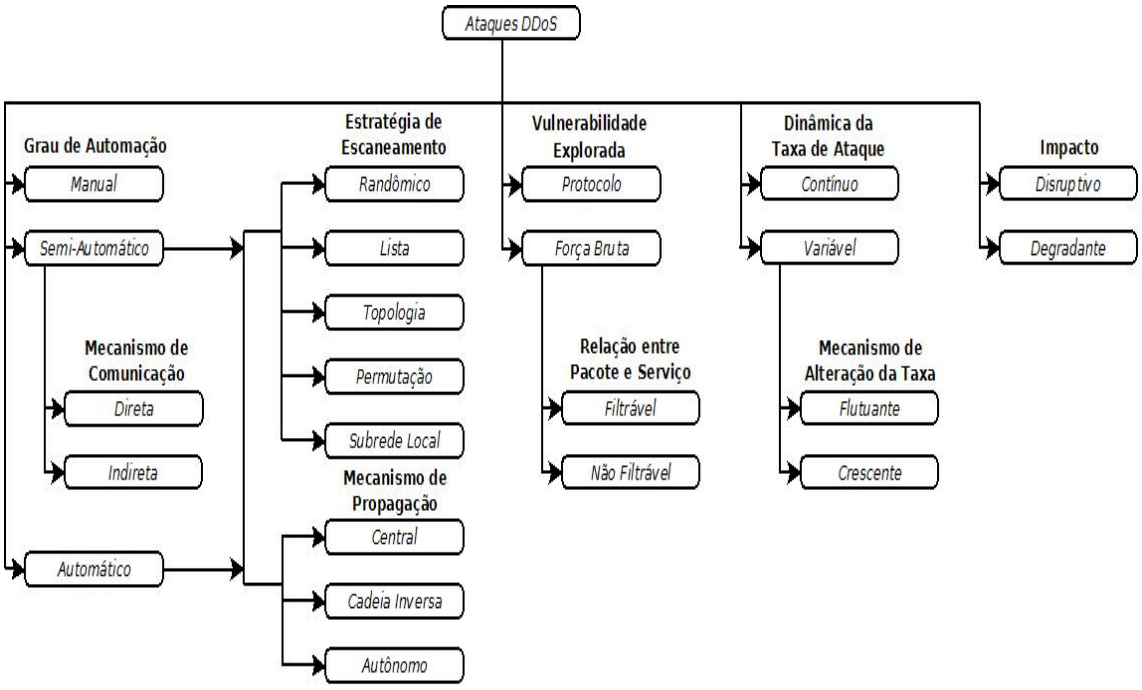


FIG. 2.2: Taxonomia de Ataques *DDoS* [14]

Para se conceber uma taxonomia de ataques *DDoS*, torna-se necessário analisar cada classe de ataque individualmente, para isso, deve-se levar em consideração três propriedades de cada tipo de ataque: [15]

- *Meios de preparação e execução do ataque:* os meios utilizados para recrutar, explorar e infectar as máquinas usadas no ataque são analisados. Ou seja, as

ferramentas e técnicas usadas nas fases de “Captura de *mestres* e *escravos*” e de “Instalação de *daemons* e *clientes*” são analisadas;
- *Características do ataque:* as particularidades da técnica do ataque, propriamente ditas, são estudadas;
- *Danos do alvo:* aqui são levantados todos os potenciais prejuízos do *alvo*.

A FIG. 2.2 apresenta a taxonomia de ataques *DDoS*.

2.2.1.1 Classificação pelo Grau de Automação

A *classificação pelo grau de automação* especifica se as fases “Captura de *mestres* e *escravos*” e “Instalação de *daemons* e *clientes*” serão feitas de forma *manual*, *semi-automática* ou *automática*.

a) Ataque Manual

Todas as três fases do ataque, detalhados na seção 3.1.2, são feitas manualmente pelo atacante. Atualmente, essa prática não é muito utilizada, visto que se pode automatizar muitos procedimentos.

b) Ataque Semi-Automático

Todo ataque *semi-automático*, para ser efetivado, precisa da comunicação entre os *mestres* e os *escravos*. Os *escravos* precisam relatar sua disponibilidade aos *mestres*, através de um canal de comunicação, e, os *mestres* também precisam desse canal para enviar suas ordens. De acordo com essa forma de comunicação os ataques *semi-automáticos* são divididos em:

- *Ataques com Comunicação Direta:* utilizam a própria rede, através de endereços IPs. Dessa forma as máquinas *mestres* ficam mais expostas, pois podem ser identificadas por *scanners* de rede;
- *Ataques com Comunicação Indireta:* utilizam canais intermediários, onde é permitido o anonimato, como, por exemplo, os canais *IRC*.

c) Ataque Automático

Neste tipo de ataque, as três fases, explicadas na seção 3.1.2, são manipuladas automaticamente, bastando um comando ser enviado do *atacante*. Dessa forma é evitado o contato entre o *atacante* e os *escravos*. Assim, a exposição do atacante é mínima.

Tanto os ataques *semi-automáticos* quanto os *automáticos* possuem *classificação por estratégia de escaneamento* e *classificação por mecanismo de propagação*.

d) Classificação pela Estratégia de Escaneamento

A *classificação pela estratégia de escaneamento* se divide da seguinte forma:

- *Ataques com Escaneamento Randômicos:* É estabelecido um espaço de endereços *IPs*. Em seguida é dado início ao escaneamento/recrutamento, e, cada *host* comprometido fica responsável por continuar com o procedimento, recursivamente, em *IPs*, escolhidos aleatoriamente, dentro do espaço de endereços já determinado.
- *Ataques com Escaneamento de Lista:* A partir de uma lista de endereços, já estabelecida, dá-se início ao escaneamento/recrutamento. Quando se identifica uma máquina vulnerável, envia-se parte da lista, para que ela fique encarregada em dar seguimento à tarefa, de forma recursiva, com tais endereços. Nesse tipo de ataque, não há colisão de endereços escaneados;
- *Ataques com Escaneamento de Topologia:* Usam informações contidas em um *host* infectado para propagar-se, como, por exemplo, *e-mails* que infectam e utilizam os endereços de contatos da vítima para se proliferar;
- *Ataques com Escaneamento de Permutação:* É estabelecido um espaço de endereços *IPs* com um índice associado. A partir disso, cada máquina comprometida recebe uma permutação diferente dos índices dos *IPs* determinados, e prossegue com o trabalho, recursivamente.
- *Ataques com Escaneamento de Subrede Local:* Pode ser implementada junto com qualquer das técnicas supracitadas. Tem por objetivo estender o procedimento à máquinas da subrede do *host* infectado.

e) Classificação pelo Mecanismo de Propagação

A *classificação pelo mecanismo de propagação* pode ser de três tipos:

- *Ataques com Fonte Central de Propagação:* O código do ataque fica armazenado em um ou mais servidores centrais. Assim que se captura um *escravo* o código é copiado dessa central.
- *Ataque com Propagação em Encadeamento Inverso:* O código do ataque é copiado da máquina responsável pelo sistema de exploração de vulnerabilidades. Assim, a máquina recém infectada se torna a fonte para o próximo passo da propagação.
- *Ataques com Propagação Autônoma:* Evita a necessidade de se copiar o código de ataque. As instruções de ataque já estão inseridas nas máquinas infectadas durante a fase de exploração.

2.2.1.2 Classificação pela Vulnerabilidade Explorada

a) Ataques de Protocolo

Exploram uma característica específica ou um erro de algum protocolo presente na máquina da vítima, com a intenção de consumir quantidades excessivas de seus recursos. Exemplificando, temos o ataques *TCP SYN Flood* (preenchendo a fila de conexões, indefinidamente), de requisições *CGI* (consumindo o tempo de *CPU*, após múltiplas requisições) e de serviços de autenticação (usando, excessivas vezes, os recursos de autenticação).

b) Ataques de Força Bruta

Caracterizam-se por iniciar uma grande quantidade de operações, de requisições de serviço, aparentemente legítimas. Se uma rede permite um volume de tráfego maior do que o volume que a vítima pode processar, essa rede se torna um facilitador para que os recursos da vítima sejam esgotados. Dependendo do conteúdo dos pacotes enviados e do serviço requisitado, da vítima, esse tipo de ataque divide-se em:

- *Ataques Filtráveis:* São formados por pacotes falsos ou pacotes para serviços de operações não críticas da vítima, assim conseguindo passar por um *firewall*. Por exemplo, ataques por inundação *UDP* ou por inundação de requisições *ICMP* em um servidor *web*;
- *Ataques Não-Filtráveis:* São formados por pacotes que requisitam recursos críticos da vítima. Diferenciam-se dos *ataques de protocolo* por necessitarem do estabelecimento de uma conexão. São exemplos de ataques os de inundação

por requisições *HTTP* e *DNS*.

2.2.1.3 Classificação pela Dinâmica da Taxa de Ataque

Podem ser ramificadas em:

a) Taxa de Ataque Contínua

É usado pela maioria dos ataques conhecidos. Após a ativação do ataque, os *escravos* enviam os pacotes que configuram o ataque, utilizando todos os recursos da máquina *atacante* para, no menor tempo possível, consumir os recursos do *alvo* e assim causar a instabilidade dos seus serviços ou até mesmo a paralização dos mesmos. Porém, isso tem a desvantagem de facilitar a detecção do ataque.

b) Taxa de Ataque Variável

Esse tipo de ataque é mais cauteloso, ele varia a taxa de ataque visando evitar a sua detecção. São diferenciados em:

- *Taxa de Ataque Crescente:* Degrada de forma crescente e lentamente os recursos do *alvo*. O tempo de ataque pode perdurar, adiando assim a sua detecção;
- *Taxa de Ataque Flutuante:* Ajusta a taxa de ataque de acordo com o comportamento do alvo, mitigando assim o efeito de sua detecção.

2.2.1.4 Classificação por Impacto

Dependendo do impacto sobre o *alvo* pode-se dividir o ataque em:

a) Ataque Disruptivo

Seu objetivo é não permitir, absolutamente, que os serviços do *alvo* sejam utilizados pelos seus clientes.

b) Ataque Degradante

Seu foco é consumir porções dos recursos do *alvo*. Por mais que esse tipo de ataque não cause o total impedimento de um serviço, ele tem a vantagem de não ser

detectado por um longo período de tempo.

2.2.2 Taxonomia de Sistemas de Defesa para ataques do tipo *DDoS*

Ataques *DDoS* têm se tornado cada vez mais frequentes e seus impactos cada vez mais graves. Com isso, nasceram vários mecanismos de defesa, cada um visando a proteção contra cada tipo de ataque. Portanto, esses sistemas de defesa tem por objetivo desmobilizar uma ou mais classes de ataques, descrita na seção anterior. A FIG. 2.3 ilustra a classificação dos mecanismos de defesa.

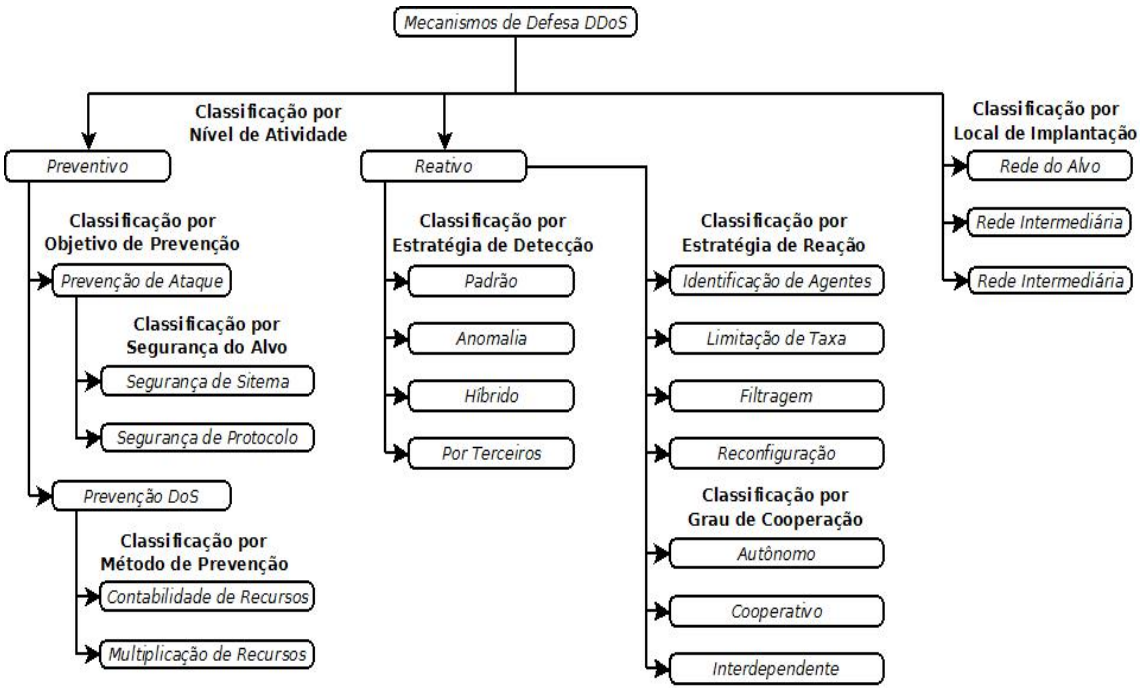


FIG. 2.3: Taxonomia de Defesa *DDoS* [14]

2.2.2.1 Classificação por Nível de Atividade

A partir do nível de atividade de um sistema de defesa *DDoS*, tem-se:

a) Mecanismos Preventivos

A finalidade desse dispositivo é tentar eliminar completamente o ataque, ou, não sendo possível, permitir que os *alvos* suportem o ataque, não negando seus serviços. De acordo com esses dois objetivos, os *mecanismos preventivos* são divididos em:

- *Prevenção de Ataque:* A configuração do sistema é alterada para eliminar a possibilidade de um ataque *DDoS*. São de dois tipos:
    - *Segurança de Sistema:* Melhoram a segurança global do sistema e previne contra intrusões e mau uso do sistema monitorando-o contra acessos não autorizados [16], removendo erros, atualizando as aplicações de segurança e fazendo uso de ferramentas de *firewall*, de anti-vírus e de detecção de intrusão [17];
    - *Segurança de Protocolo:* Está focado no problema de protocolos que foram mal projetados ou mal implementados. Para resolver esse tipo de dificuldade pode-se, por exemplo, forçar uma autenticação do cliente antes de ele usar um recurso de alto custo do servidor, podendo ser implementado por um método de avaliação da rede [18] ou até mesmo por um protocolo [19];
- *Prevenção de Negação de Serviços:* Permite que o *alvo* resista ao ataque sem negar seus serviços a usuários legítimos. Essa prevenção pode ocorrer de duas formas:
    - *Contabilidade de Recursos:* Fiscaliza o acesso de cada cliente aos recursos, baseando-se em privilégios do usuário e no comportamento de cada usuário. Esse mecanismo pode ser implementado através da utilização da criptografia [20], da criação de métodos próprios [21] [22] [23] ou da criação de regras *QoS* [24];
    - *Multiplicação de Recursos:* Aumenta a capacidade dos recursos para suportar os ataques *DDoS*. Um exemplo simples são os sistemas que possuem um conjunto de servidores com balanceamento de carga e com conexão de alta velocidade.

b) Mecanismos Reativos

Procura aliviar o impacto no *alvo*, causado por um ataque. Para isso, é necessário detectar o ataque, o mais cedo possível e com um baixo grau de falsos positivos, e então, caracterizar o ataque, a partir de uma inspeção nos pacotes pertencentes ao fluxo, para, enfim, determinar qual o mecanismo a ser usado para reagir ao ataque. Baseado na estratégia de detecção do ataque são divididos em mecanismos que implementam:

- *Detecção de Padrões de Ataque:* Nesta categoria os mecanismos armazenam modelos de ataques conhecidos em um banco de dados. Cada comunicação é monitorada e comparada com os modelos, existentes no banco de dados, para se descobrir se a dada comunicação é um ataque ou não. A desvantagem desse

método é sua limitação a ataques conhecidos, por seu banco de dados. Porém, o mecanismo permite que modelos de novos ataques sejam adicionados;

- *Detecção de Anomalia de Ataque:* Gera um modelo do tráfego esperado no sistema. Esse modelo é, periodicamente, comparado com os modelos capturados em tempo de execução, para detectar alguma anomalia. Exemplos desse mecanismo são descritos em [25] através de seu sistema de defesa distribuída, [26] usando mensagens de *Pushback* para controlar agregados na rede, [27] que é é um sistema de configuração rápida, implantação e gestão da superposição das camadas protetoras que visam tanto pró-ativa e reativamente resistir os ataques do tipo *DDoS*, [28] com uma estrutura de dados própria para detecção de ataques e [29] que propõe um *Firewall* reverso. Esse sistema possui a vantagem de detectar ataques desconhecidos. São focados em duas questões:
  - *Limiar Fixo:* As anomalias são detectadas quando o modelo capturado e o modelo esperado do sistema diferem de um valor acima de um limiar determinado. Um limiar baixo causa muitos falsos positivos, enquanto que um limiar alto pode deixar de detectar alguns ataques;
  - *Atualização de Modelo:* Captura, automaticamente, novos modelos de comportamento esperados pelo sistema, a partir de estatísticas geradas, durante o período em que ataques não são detectados. Porém, essa característica torna o mecanismo vulnerável a investidas com *taxa variável de ataque*.
- *Detecção Híbrida de Ataque:* Combina detecção por padrões e por anomalias, usando o sistema de detecção de anomalias para descobrir e atualizar o banco de dados com os novos ataques. A geração dos novos modelos de ataques é a parte crítica desse mecanismo, pois pode ser gerado um modelo enquadrado como de comportamento normal, sendo que, na realidade, neste há um ataque inserido;
- *Detecção de Ataque de Terceiros:* Esse sistema não possui um mecanismo próprio de detecção dependendo de mecanismos de rastreamentos externos que sinalizam a ocorrência do ataque e provêem a sua caracterização, como sistemas de marcação e autenticação avançados para *IP Traceback* [30], uma abordagem algébrica para *IP Traceback* [31], mensagens *ICMP Traceback* [32], um suporte de rede para *IP Traceback* [33] ou um procedimento *Hash-Based* de *IP Traceback* [34].

O objetivo da resposta a um ataque é mitigar o impacto do ataque sobre o *alvo*, impondo o mínimo dano aos usuários legítimos da vítima [14]. Esses *mecanismos reativos* são caracterizados pela *estratégia de resposta*, que pode ser:

- *Identificação de Agentes:* Proporciona ao *alvo* dados sobre as máquinas que estão efetuando o ataque. Essas informações podem ser combinadas com outras técnicas para reduzir o efeito do ataque. Esse mecanismo é encontrado em diversas técnicas de rastreamento, como as citadas no tópico *Detecção de Ataque de Terceiros*, e técnicas de eliminação de *spoofing* como em [35] e [36] nas quais seus protocolos empregam o endereço *IP* de origem para a validação do agente;
- *Limitação de Taxa:* Determina um limite para a velocidade de conexão das máquinas detectadas como invasoras, por algum mecanismo de detecção. Normalmente, é usado quando a detecção possui alto nível de falsos positivos.
- *Filtragem:* Usa os resultados de um mecanismo de detecção para bloquear totalmente o acesso do *atacante*. Deve ser escolhido um mecanismo que não gere falsos positivos, para que um usuário legítimo não seja impedido de usar algum recurso;
- *Reconfiguração:* Altera-se a topologia do *alvo* ou da rede intermediadora [37] [27], para adicionar mais recursos para o *alvo* [25] ou para isolar as máquinas de ataque.

Mecanismos de defesa reativos podem executar a detecção e reagir isoladamente ou cooperativamente com outras entidades na *Internet*. Baseando-se na ordem de cooperação, tais mecanismos são diferenciados entre:

- *Autônomo:* Executa a detecção e a reação de forma independente. *Firewalls* e sistemas de detecção de intrusão são exemplos desse mecanismo;
- *Cooperativo:* Pode se comportar como um *mecanismo autônomo*, mas é capaz de atingir melhores resultados através da cooperação com outras entidades. Detecta a existência de ataques *DDoS* através da congestão de roteadores, e, age localmente limitando a velocidade do tráfego;
- *Interdependente:* Esses mecanismos dependem de outras entidades para detectar ou para reagir com eficiência. Técnicas de rastreamento são exemplos de mecanismos interdependentes.

2.2.2.2 Classificação por Local de Implantação

A localização dos mecanismos de defesa podem ser na:

- *Rede do Alvo:* Implanta os sistemas de defesa na rede da vítima;
- *Rede Intermediária:* Os mecanismos de defesa são instalados em uma rede, com infraestrutura capaz de suportar grande quantidade de acessos;
- *Rede Origem:* Instala os sistemas de defesa na rede dos clientes. Assim, previne que as redes dos usuários sejam fonte de ataques *DDoS* [29].

2.2.3 Aplicação das Taxonomias

Neste trabalho foram realizados ataques *online* e *offline*, que podem ser analisados posteriormente. O conjunto de ataques efetuados permitiu abordar as seguintes características da taxonomia de ataque:

- Grau de Automação: manual e semi-automático;
- Estratégia de Escaneamento: randômico, subrede local;
- Vulnerabilidade Explorada: protocolo e força bruta;
- Relação entre Pacote e Serviço: filtráveis e não filtráveis;
- Dinâmica da Taxa de Ataque: contínuo;
- Impacto: disruptivo.

Como o número de computadores necessários para realizar um ataque real do tipo *DDoS* é inviável, neste trabalho, todos os sistemas de proteção foram desabilitados durante os ataques *online*. Portanto, em relação à taxonomia de defesa não foi aplicado nenhum de seus conceitos, visando atingir o máximo potencial de ataque possível.

2.3 Experimento *DARPA*

Em 1998, nos *Laboratórios Lincoln* do *MIT*, foi gerada, pelo *DARPA*, uma base de dados de detecção de intrusão, chamado de *KDD-98* [38].

Para a criação dessa base, foram executadas avaliações *off-line* e avaliações em tempo real. A avaliação realizada *off-line*, a mais complexa, foi uma tentativa inicial de se aplicar a tecnologia de detecção de intrusões, e, analisar seus resultados.

Os objetivos do experimento eram limitados, pois foi projetado, apenas, para:

- Avaliar os sistemas de detecção de ataques do *DARPA*;
- Medir as taxas de falsos positivos, usando um tráfego de fundo semelhante ao comportamento da rede da base da Força Aérea dos Estados Unidos;
- Mensurar as taxas de detecção de ataques, iniciados remotamente contra *hosts UNIX*.

Dois principais grupos de pesquisa participaram das avaliações do experimento:

- *MIT Lincoln Laboratory:* Encarregou-se da linha de avaliação *off-line*, incluindo-se semanas de treinamento e tráfego de teste com mais de 300 casos de 38 tipos de ataques. Resultou na, supracitada, base *Kdd-98*;
- *AFRL:* Responsabilizou-se pela avaliação em tempo real, usando um número menor de sistemas, uma rede mais complexa, menos ataques e quatro horas de

tráfego.

Os resultados das avaliações realizadas em 1998 ajudaram a descobrir os pontos fortes e fracos das abordagens de mecanismos de detecção, e, tiveram, também, uma forte influência nos objetivos da pesquisa de detecção de intrusões do *DARPA*. Porém, por maiores avanços que as avaliações tenham ocasionado, a base é ultrapassada e não representa a *Internet* atual [39]. Indo mais além, pode-se afirmar que nenhuma base de dados que esteja disponível na internet, quer seja agora, ou daqui a 10 anos irá refletir o comportamento da rede. Um motivo visível para essa estimativa é a mudança constante do perfil da rede, dada a quantidade de novas funcionalidades que estão sendo criadas. Serviços como, por exemplo, o *orkut* (criado em 2004) e o *twitter* (criado em 2006) mudam a característica geral da rede. Outras funcionalidades irão surgir e, com isso, as bases de dados não podem ser estáticas.

3 DESCRIÇÃO DO ATAQUE ESCOLHIDO

Dentre tantas outras ofensivas de ataque, foi selecionado o ataque *7.7 DDoS* devido, principalmente, a sua magnitude. A primeira grande ofensiva deste ataque ocorreu em 4 de Julho de 2009 quando uma série de *websites* comerciais e do Governo dos EUA e, posteriormente, a *websites* Sul-Coreanos começaram a sofrer ataques de negação de serviço com as mesmas características [40]. A análise do tráfego de ataque apresentou informações presentes na tabela a seguir.

Tipo	Outras Informações
ICMP Echo Request Flood	
UDP Flood	Porta do alvo: 80
TCP SYN Flood	Porta do alvo: 80
IP Protocol 0 Flood	
HTTP GET Request Flood	Requisição da pasta raiz: "/"

TAB. 3.1: Tabela de Ataques [40]

Nas próximas seções serão descritas características e particularidades desse ataque.

3.1 Classificação do Ataque

De acordo com a taxonomia adotada neste trabalho, o ataque em questão apresenta um grau de automação semi-automático com comunicação indireta devido aos níveis de servidores distribuídos de comando e controle. Quanto a propagação, o *malware* apresentou disseminação essencialmente central, com repositórios em servidores de códigos ou *web sites* explorados. As estratégias de ataque implementadas exploraram vulnerabilidades de protocolo, como *TCP SYN Flood*, e também força-bruta, como *HTTP Get Flood*, apresentando taxa de ataque constante. O impacto do ataque pode ser considerado perturbador já que causou a negação de serviço de diversos *websites* mas, vale notar que, alguns web sites tiveram seu desempenho apenas degradado.

3.2 Detalhamento do Ataque

3.2.1 *ICMP Echo Request Flood*

Este ataque se dá através do envio de pacotes *ICMP Echo Request* para a vítima sendo que essa, por sua vez, retorna um pacote *ICMP Echo Response* para o atacante. A negação do serviço ocorre devido ao consumo de recursos da *CPU* da

vítima, envolvido nesse processo de requisição e resposta, ou através do consumo de largura de banda devido à quantidade de pacotes *ICMP* trafegados na rede. Enquanto são utilizados pacotes *ICMP* maiores para causar negação através do consumo de recursos, pacotes *ICMP* menores são utilizados para sobrecarregar o roteamento e, consequentemente, consumir largura de banda.

3.2.2 *UDP Flood*

O ataque *UDP Flood* é semelhante ao ataque *ICMP Request Flood*, porém são enviados pacotes *UDP* para porta de destino. Assim como no *ICMP Request Flood*, a vítima irá processar o pacote *UDP* e irá retornar pacotes *ICMP Echo Response*. O envio desses pacotes como resposta também poderia agir como um ataque de negação de serviço para o atacante, por isso utiliza-se a técnica de *IP Spoofing* para alterar o endereço *IP* de origem para livrar-se do tráfego de resposta.

3.2.3 *TCP SYN Flood*

O ataque *TCP SYN Flood* utiliza-se de uma vulnerabilidade do protocolo *TCP/IP* que permite que um atacante esgote os recursos da vítima através de conexões *TCP* semi-abertas. A conexão *TCP* acontece em três etapas (*three-way handshake*). Um cliente que deseja iniciar uma conexão em determinada porta aberta pelo servidor, no caso a vítima, envia um *SYN*. O servidor, por sua vez, responde com um *SYN-ACK* e para finalizar o processo de estabelecimento de conexão o cliente envia um *ACK* e a conexão estará pronta para transferência de dados. O ataque *TCP SYN Flood* ocorre quando um atacante inicia diversas conexões *TCP* sem, no entanto, enviar o *ACK*, obrigando o servidor a aguardar a finalização do processo. Como cada porta *TCP* (*backlog*) possui um número limitado de conexões semi-abertas, o servidor ignora novas requisições de conexão até que o *timeout* das conexões as invalide, assim é criado um período de negação de serviço.

Da mesma maneira como os outros ataques mencionados, este ataque utiliza-se da técnica de *IP Spoofing* de modo a evitar alta carga de pacotes *SYN-ACK*. No entanto, a escolha do *IP* deve ser feita garantindo que o endereço seja roteável e inalcançável. Dessa maneira o servidor será notificado através de pacotes *ICMP* de que o cliente encontra-se inalcançável, porém a camada *TCP* irá ignorar esses avisos. Caso ele seja alcançável, o cliente receberá o pacote *SYN-ACK* e, por sua vez, responderá com pacotes *RST* liberando a conexão semi aberta.

3.2.4 *HTTP GET Request Flood*

HTTP GET Request Flood é utilizado como ataque complementar para a sobrecarga do *Web Server*. Essencialmente, a requisição *HTTP* é semelhante à de um usuário legítimo, porém é executada diversas vezes e tem como objetivo consumir recursos da vítima.

3.2.5 *Protocol 0 Flood*

O modelo *TCP/IP* define um campo "*protocol*" no cabeçalho *IP* responsável por informar qual protocolo da camada de transporte encapsula o *payload*. Quando um protocolo não disponível na implementação da camada de transporte é referenciado, o *host* de destino gera um pacote *ICMP* tipo 3, *Destination Unreachable*, com código 2, *Protocol Ureachable Error*, informando a impossibilidade de processamento do pacote pela camada de transporte.

Essa característica é explorada no ataque de *IP Protocol Flood* de forma semelhante ao *ICMP Flood*. A demanda de processamento na recepção do pacote *IP* e na criação do pacote *ICMP Protocol Unreachable Error* permite o esgotamento de recursos da vítima. O atacante inunda a vítima com pacotes *IP* com campo "*protocol*" com um valor não implementado. Esse valor pode ser escolhido entre protocolos pouco utilizados ou através de ferramentas que varrem os protocolos disponíveis na vítima.

4 EXPERIMENTOS

4.1 Experimento Realizado

Como dito anteriormente, o objetivo do trabalho é fornecer uma base de dados de um ataque e preparar uma estrutura simples para servir para testes futuros de outros ataques. Sendo assim, o experimento realizado foi a implementação dos algoritmos de ataques que compõem o ataque escolhido, descrito na seção anterior, para gerar o *trace* bem como montar uma estrutura que seja de fácil atualização e reuso para aplicar outras abordagens de ataques.

Primeiramente foi montada uma estrutura orientada a objetos escrita na linguagem *Java* para permitir a realização de ataques de forma *online*, ou seja, atacar uma maquina em tempo real. Para isso, foram criadas quatro classes dentro do pacote "ataques": *HttpRequestFlood.java* (FIG. 4.1), *ICMPRequestFlood.java* (FIG. 4.2), *SYN_Flood.java* (FIG. 4.3) e *UDP80 Flood.java* (FIG. 4.4).

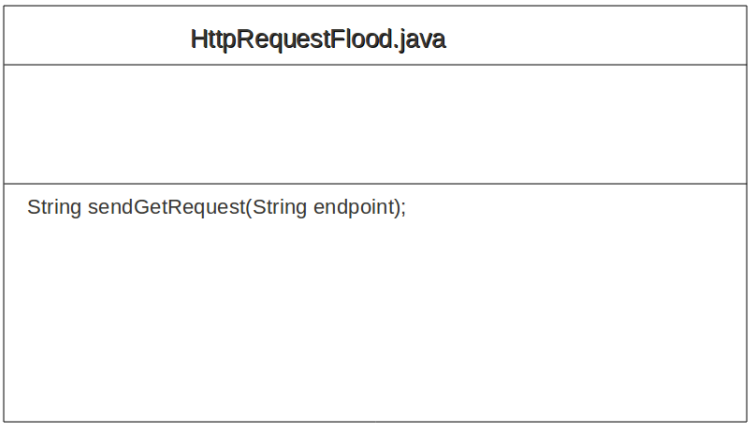


FIG. 4.1: Classe HttpRequestFlood

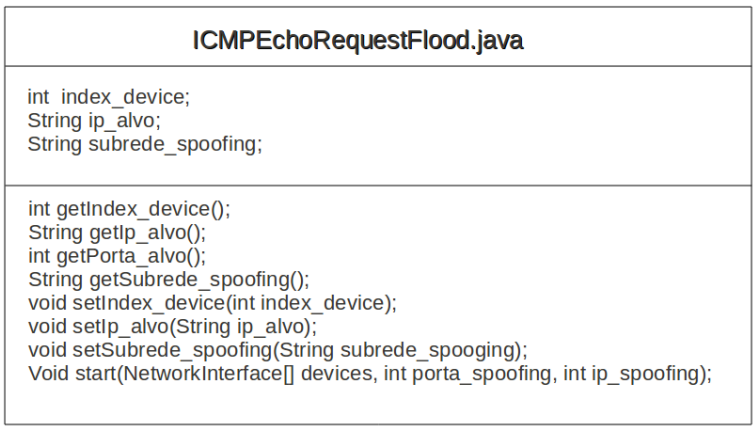


FIG. 4.2: Classe ICMPEchoRequestFlood

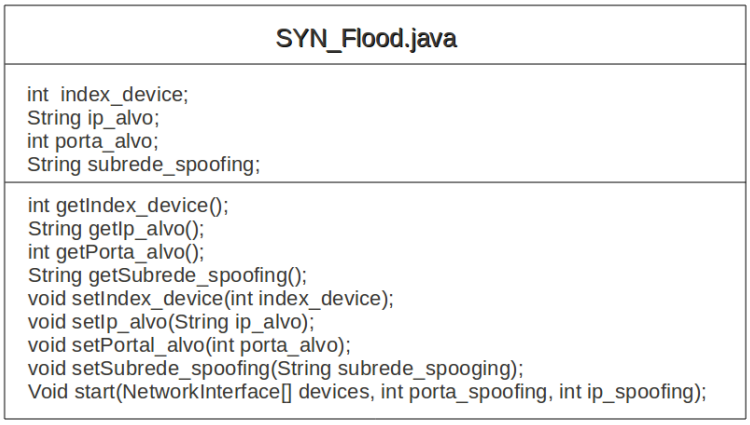


FIG. 4.3: Classe SYNFlood

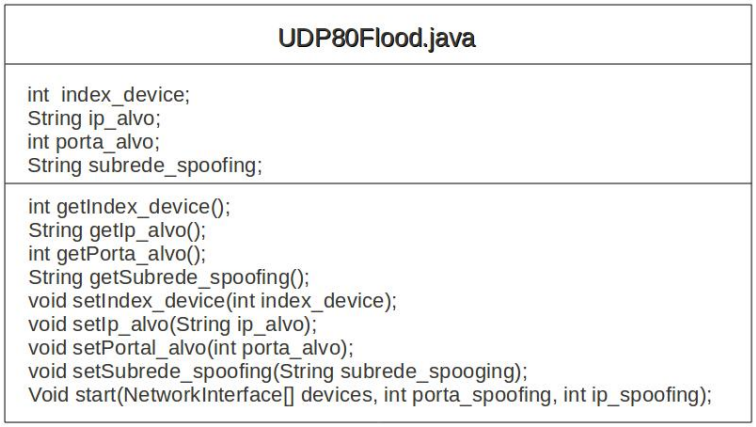


FIG. 4.4: Classe UDP80Flood

A implementação das três últimas classes foi feita utilizando a biblioteca *Jpcap* [41], uma biblioteca de *Java* utilizada para captura e envio de pacotes de rede baseada na ferramenta *WinPcap*. *WinPcap* consiste em um driver e uma biblioteca que permitem ao sistema operacional acesso às camadas de rede e que aplicações façam a captura e transmissão de pacotes utilizando determinados protocolos [42]. Esta ferramenta possui alguns recursos interessantes como a filtragem de pacotes, provê estatísticas da rede e dá suporte para captura remota de pacotes. Semelhante a essa ferramenta, existe a *LibPcap* que diferentemente da *WinPcap*, não é comercial e sim *open source*. Dentre os principais recursos da biblioteca *Jpcap* podemos citar:

- Interfaces de redes disponíveis no sistema;
- Capturar pacotes da interface de rede;
- Aplicar filtros à captura de pacotes de rede;
- Gravar pacotes de rede capturados em arquivos;
- Ler pacotes de rede gravados em arquivos.

A classe *ICMPRequestFlood.java* tem o objetivo de simular o envio de um pacote *ICMP* e conta com os campos:

- int index_device: indica qual a placa de rede a máquina atacante utiliza para inundar a máquina atacada;
- string ip_alvo: guarda o endereço de *IP* da máquina atacada;
- string subrede_spoofing: guarda a subrede utilizada para realizar o *spoofing*.

A classe *SYN_FLood.java*, responsável por simular o envio de um pacote *UDP* e *UDP80Flood.java*, responsável por simular o envio de um pacote *UDP*, possuem os mesmo campos da classe *ICMPRequestFlood.java* com adição de um outro campo:

- int porta_alvo: indica qual porta do alvo será atacada.

A classe *HttpRequestFlood* possui a definição do método *sendGetRequest* onde está implementada a leitura de uma conexão feita com uma *URL* passada como parâmetro.

As classes *ICMPRequestFlood.java*, *SYN_Flood.java* e *UDP80Flood.java* contém a definição do método "*start*" onde está implementado o envio de pacotes *ICMP*, *TCP* e *UDP* respectivamente. Essa função permite que seja realizado o *IP spoofing*, isto é, permite que seja escolhido um *IP* diferente do *IP* da própria máquina do atacante para ser o *IP* de origem do pacote enviado. Os pacotes *TCP* e *UDP* também permitem o *spoofing* da porta utilizada. Essa característica de *spoofing* é interessante para esconder a autoria do ataque, bem como aumentar o número de requisições ao serviço da vítima e evitar o tráfego de retorno dos pacotes *ACK*.

Essas quatro classes serviram como base para a criação de outras cinco classes contendo os métodos "*main*" que representam os cinco ataques característicos do ataque selecionado para o trabalho. Essas cinco classes são:

- *Main_HttpRequestFlood.java*
  - Representa o ataque *HTTP Request Flood*
  - Utiliza a classe *HttpRequestFlood.java*
- *Main_TCPSYN.java*
  - Representa o ataque *TCP SYN Flood* - porta 80, com *spoofing* de *IP*
  - Utiliza a classe *SYN_Flood.java*
- *Main_ICMP.java*
  - Representa a o ataque *ICMP Echo Request Flood*, com *spoofing* de *IP*
  - Utiliza a classe *ICMPRequestFlood.java*
- *Main_UDP80*
  - Representa o ataque *UDP 80 Flood*, com *spoofing* de *IP*
  - Utiliza a classe *UDP80Flood.java*
- *Main_IPProtocol.java*
  - Representa o ataque *IP Protocol* 0 *Flood*, com *spoofing* de *IP*

- Utiliza as classes *ICMPRequestFlood.java*, *SYN_Flood.java* e *UDP80Flood.java*

Essas classes são responsáveis por gerar o *spoofing* de *IP*, e porta para o caso de pacotes *UDP* e *TCP*, além de receber como argumento o *IP* e porta do alvo. Esse *spoofing* é feito através de um *loop* de porta e posteriormente de *IP*, para cada porta e *IP* diferentes é feita uma chamada da função "*start*" de cada classe base.

5 ATAQUE *ONLINE*

Com a estrutura montada e preparada para realizar os cinco ataques que compõe nosso ataque principal, realizou-se o experimento. Os cinco ataques foram feitos separadamente utilizando uma máquina atacante e outra atacada. Os computadores estavam conectados na mesma rede através de um *switch* como ilustra a FIG. 5.1.

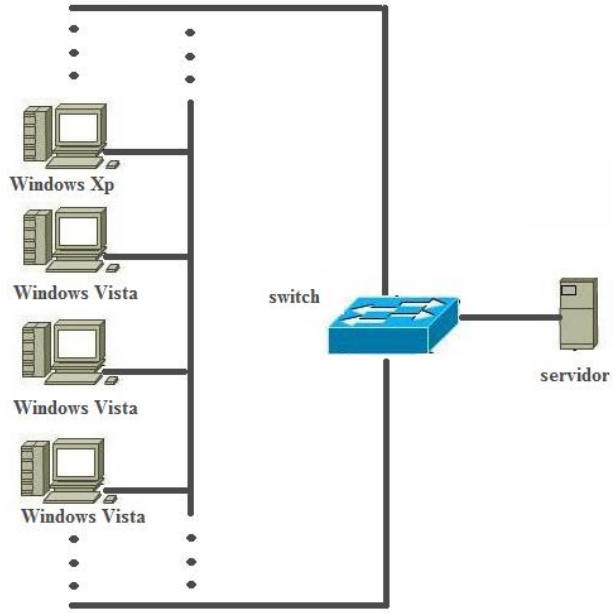


FIG. 5.1: Topologia do experimento

Durante a realização dos ataques foi utilizado o programa *WireShark* para coletar dados da rede. O *WireShark* é uma ferramenta sucessora do *Ethereal* desenvolvida para analisar os pacotes trocados pelos protocolos da rede. Desenvolvido em código aberto, o *WireShark* também conta com filtros de protocolos, permitindo ao usuário receber apenas as informações que deseja. Além disso, a ferramenta está preparada para salvar um registro das operações de análise. O aplicativo tem um *design* bem funcional e oferece recursos para o usuário customizar a interface, facilitando sua utilização. Esse programa gerou arquivos pcap contendo todos os pacotes que trafegaram na rede durante cada ataque separadamente. Além dos *traces* de ataque, foi gravado o tráfego de pacotes, sem a ocorrência de um ataque, de uma máquina da rede interna do Laboratório de Redes do Instituto Militar de Engenharia durante cerca de 20 minutos. Este *trace* foi gravado como um arquivo *pcap* (*base_normal.pcap*).

5.1 Ambiente dos Ataques DOS

Durante os experimentos foi possível perceber que, dependendo do sistema operacional, o ataque *SYN Flood* não foi possível de ser executado. Isso se deve ao

fato de sistemas operacionais com *Linux*, *Windows Vista* contarem com uma proteção *SYN Cookie* que evita a o *soopifing* de *IP* e assim impede que o *SYN Flood* cause uma negação de serviço.

Os ataques de negação de serviço simples utilizados para gerar as bases de ataque de entrada para o algoritmo *TraceMerger*, foram realizadas nas seguintes condições:

- ambiente *Windows XP* na máquina atacada e *Windows Vista* nas máquinas atacantes;
- o *firewall* dos sistemas operacionais das máquinas atacadas estavam desligados;
- rede local, sem acesso à *Internet*, com máquinas conectadas por cabos ao roteador;
- uso do roteador *Linksys* WRT54G-LA 802.11g 54Mbps com configurações de segurança desativadas.

5.2 Recursos de máquina durante um ataque de negação de serviço

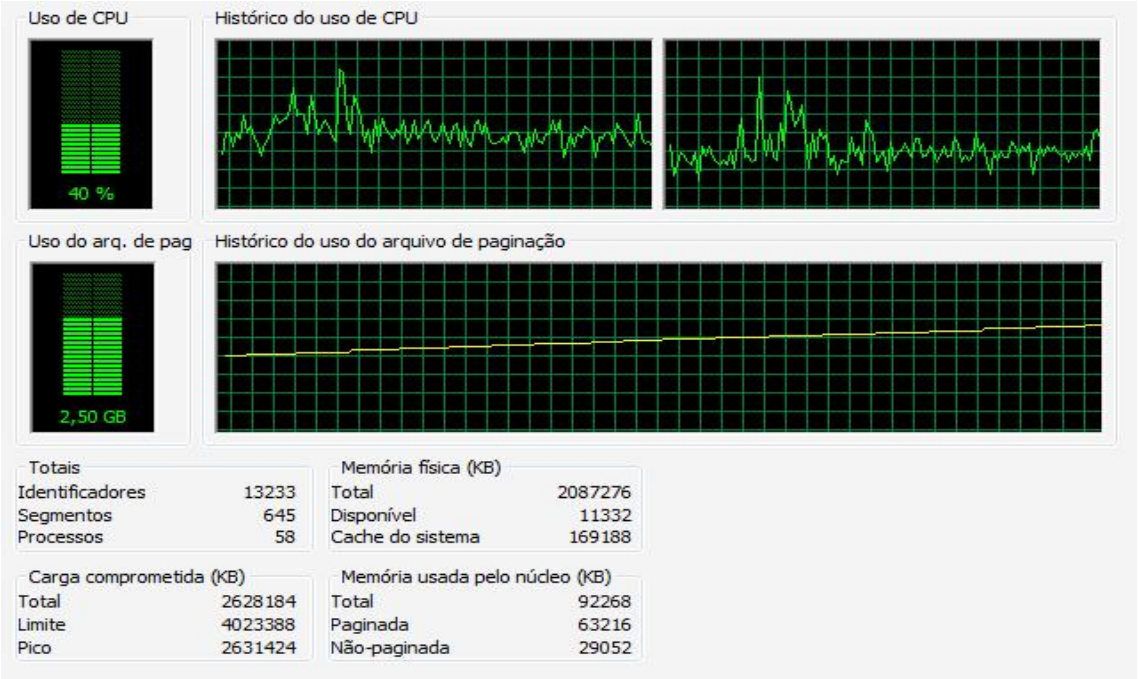


FIG. 5.2: Histórico de performance de *CPU* e memória durante um bombardeio de pacotes.

Analisando o medidor de desempenho do sistema operacional da máquina atacada para verificar o desempenho durante os bombardeamentos de pacotes pelo ataque *tcp syn flood* dos experimentos de ataques, foi verificado que ocorreu um consi-

derável aumento de paginações da memória principal (veja a FIG. 5.2), enquanto o uso corrente do processador não sofreu mudanças drásticas assim como os recursos disponíveis da placa de rede, que sofreram variações na ordem de 2,5% de uso da capacidade total conforme é observável na FIG. 5.3 Deve-se lembrar que os ataques a que este capítulo se refere não são distribuídos e que a velocidade de envio de pacotes destes experimentos são dependes da performance da máquina virtual *Java* (*JVM*).

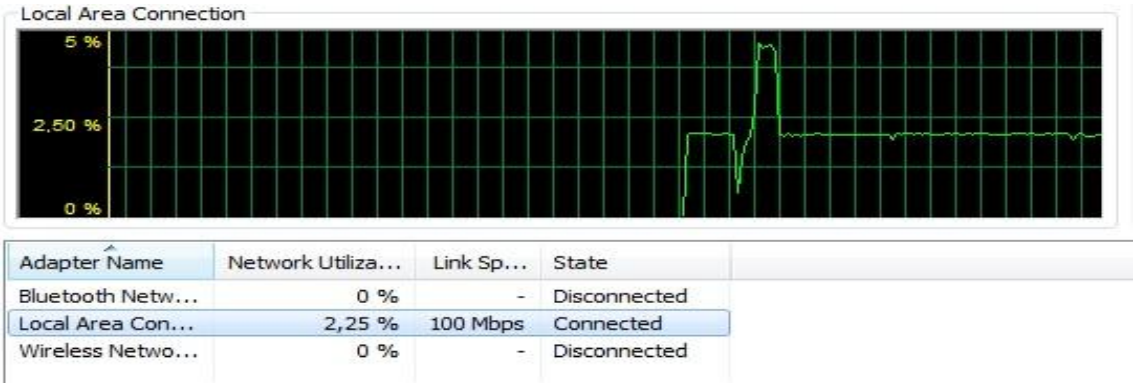


FIG. 5.3: Histórico de recursos da placa de rede durante um bombardeio de pacotes.

Chegou-se a um resultado onde é possível utilizar ataques de negação de serviço, se explorados de maneira eficiente (por exemplo, torná-los independentes da *JVM*) e com o uso de vários atacantes, para tentar privar os recursos de memória da máquina alvo, em vez de somente focar no consumo de banda de transmissão ou explorar a capacidade limite da placa de rede.

6 ATAQUE *OFFLINE*

Os resultados apresentados nesta seção são produto de uma intercalação simples entre os pacotes de dois *traces*, sendo um deles o que representa o comportamento normal da rede e o outro correspondendo ao ataque. Com isso, não era considerada as perdas de pacotes devido à expiração de seus respectivos *timeouts*.

6.1 Análise gráfica do ataque *TCP SYN Flood*

A estratégia de análise adotada nesse trabalho foi a de soma cumulativa. A análise se baseia nos intervalos de tempos em que determinados tipos de pacotes trafegam pela rede. Cada tipo de pacote representa uma *flag TCP* ativada.

Para gerar estes gráficos, que serão vistos em seguida, foram obtidos, em primeiro lugar, *traces* filtrados, tendo como regra de filtragem as *flags* desejadas. A partir disso, foram extraídos as informações relevantes desse *trace*. O *trace* de antes do ataque representa um fluxo de pacotes em uma máquina no estado de uso da *Internet* sem estar sofrendo qualquer tipo de ataque. O *trace* de ataque foi gerado a partir de um ataque real (*online*) e contém apenas os pacotes referentes ao ataque. O *trace* depois do ataque corresponde ao *trace* final, gerado pela intercalação do *trace* de ataque com o *trace* de antes do ataque.

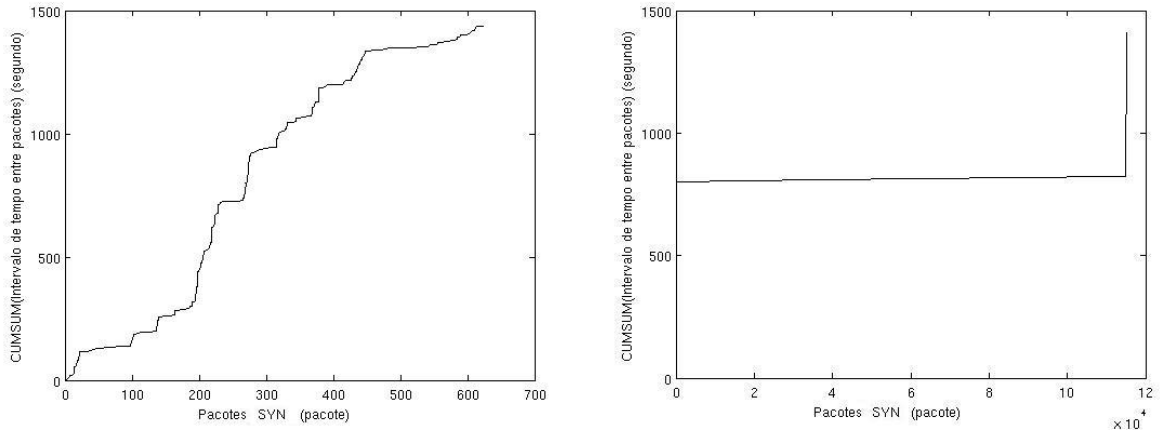
Os tipos de pacotes estudados são quatro, e, logo após procede a análise final de pacotes de todos os tipos do *trace* final de uma máquina atacada.

6.1.1 Pacotes com a *flag SYN* ativada

Na FIG. 6.1 (a) é visto um comportamento normal de uso da *Internet*. Já na FIG. 6.1 (b) temos um *trace* dos pacotes *SYN* com o ataque realizado.

Após o ataque é visível a concentração de envio em pequeno período de tempo de pacotes *SYN*, o que nesse caso controlado personifica o ataque, porém, não é possível afirmar que é uma caracterização de ataque, sempre que houver esse tipo de comportamento gráfico. Por exemplo, é possível ter-se o mesmo gráfico em uma rede que acaba de entrar em um momento de pico de acessos a *Internet* por parte dos próprios usuários da rede.

Uma sugestão para a detecção do ataque a partir de uma análise gráfica é partir para o estudo da angulação que a parte plana do gráfico faz com o eixo das abscissas.

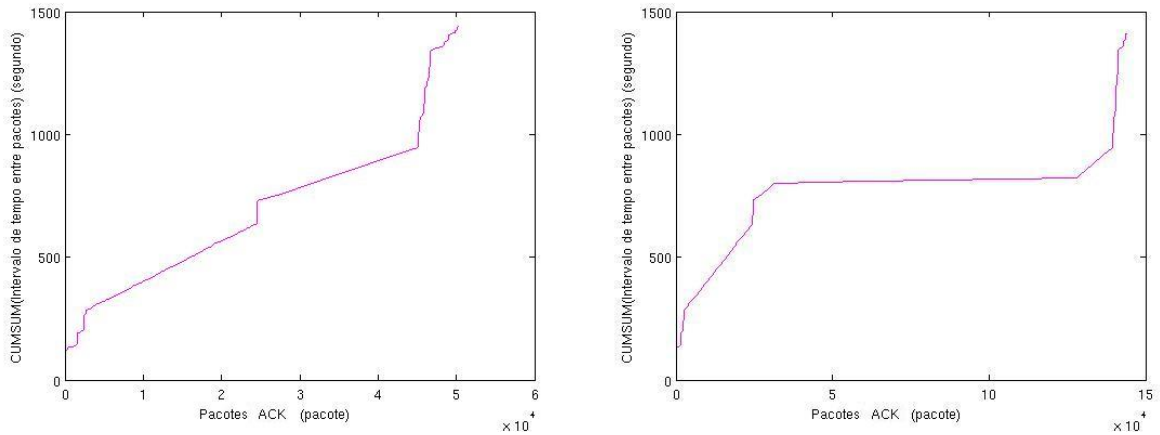


(a) Antes do Ataque (b) Após o Ataque

FIG. 6.1: *Flag SYN.*

6.1.2 Pacotes com a *flag ACK* ativada

A FIG. 6.2 nos mostra o comportamento de um *trace* de um computador que foi atacado. No eixo das abscissas temos todos os pacotes que possuem a *flag ACK* ativada. Vale lembrar que pacotes desse tipo se encontram no início de conexão *TCP* como resposta a pacotes com somente a *flag SYN* ativada. Relembrando, essa resposta é um pacote com as *flags SYN* e *ACK* ativadas. E, no eixo das coordenadas temos a soma cumulativa dos intervalos desses pacotes.



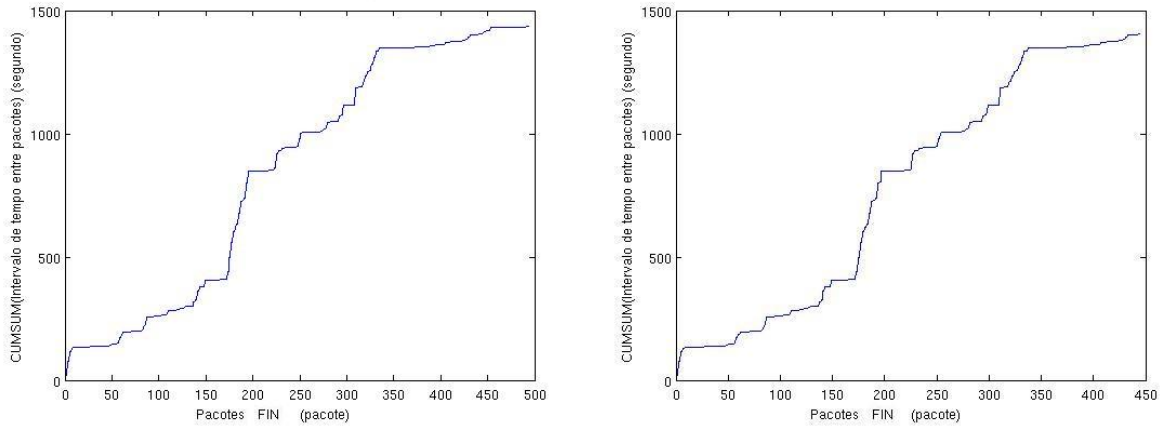
(a) Antes do Ataque (b) Após o Ataque

FIG. 6.2: *Flag ACK.*

Verifica-se que após o ataque, entre, aproximadamente, os pacotes *ACK* de número 25000 e 125000, os intervalos de tempo de um pacote para o outro se tornaram

muito pequenos, o que é conseqüência de muitas respostas de pedidos de conexão *TCP*.

6.1.3 Pacotes com a *flag FIN* ativada

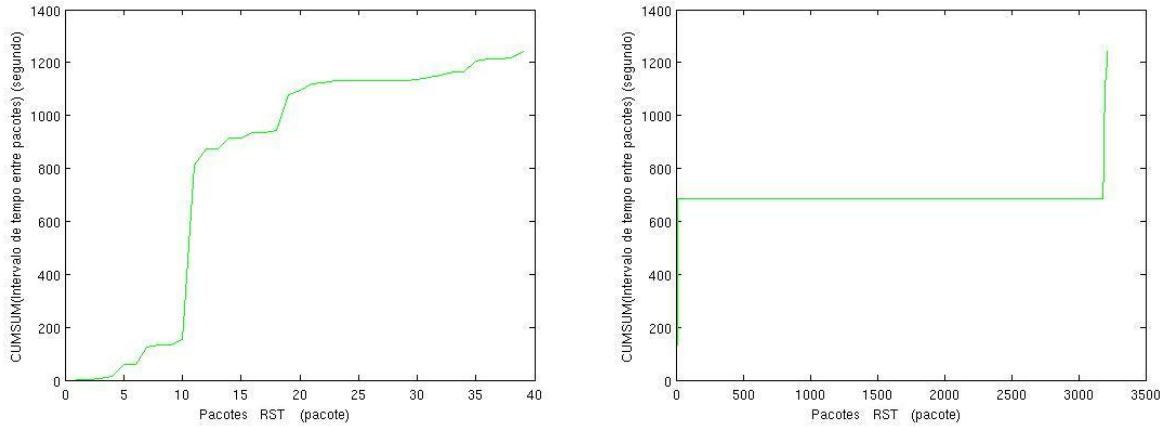


(a) Antes do Ataque (b) Após o Ataque

FIG. 6.3: *Flag FIN.*

Ao se observar a FIG. 6.3 registra-se mudanças insignificativas de comportamentos entre os gráficos (a) e (b).

6.1.4 Pacotes com a *flag RST* ativada



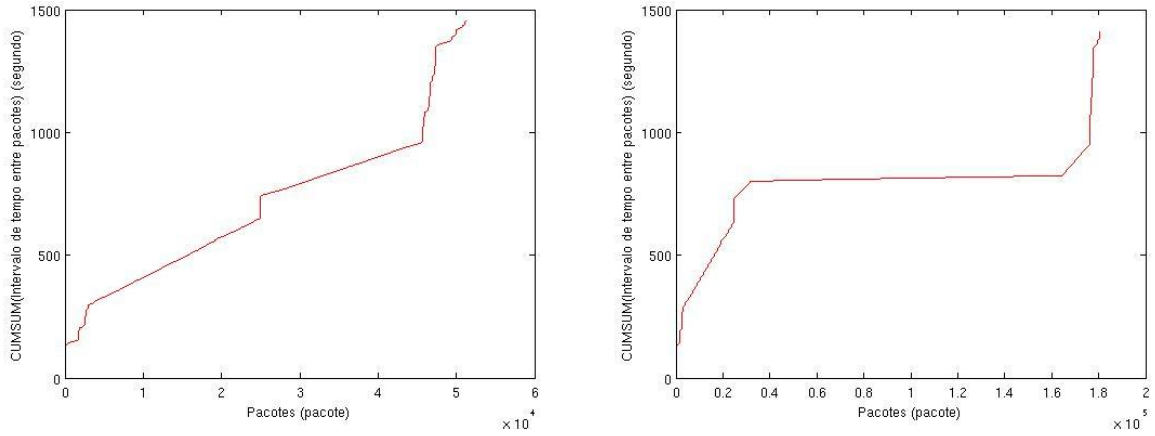
(a) Antes do Ataque (b) Após o Ataque

FIG. 6.4: *Flag RST.*

A FIG. 6.4 possui um comportamento bastante similar ao da FIG. 6.1. Nesse caso controlado isso ocorre pois para cada pacote *SYN* que expira o seu *timeout* é enviado um pacote com a *flag RST* ativada.

6.1.5 Análise geral

A FIG. 6.5 representa o *trace* sem filtro de *flags*, ou seja, todos os pacotes são representados no eixo das abscissas.



(a) Antes do Ataque (b) Após o Ataque

FIG. 6.5: *Trace.*

É possível verificar o impacto que o ataque causa na distribuição dos intervalos de tempo das transmissões dos pacotes. O gráfico tende a ter um comportamento planar. Porém, como já dito anteriormente, não é possível, fora desse caso controlado, afirmar que é possível detectar um ataque do tipo *TCP SYN Flood* a partir dessa análise gráfica. Permanece, portanto, a mesma sugestão supracitada na seção 5.1.1.

6.2 Ferramenta

Visando corrigir a deficiência encontrada no algoritmo anterior foi desenvolvida uma ferramenta que possibilita a geração de traces atacados, a partir dos cinco tipos de ataques aqui implementados, e que, também, trata as exceções inerentes ao protocolo de início de conexão *TCP*.

Uma vantagem do aplicativo é a possibilidade de extensão tanto dos tipos de ataques, quanto o número de vezes que um mesmo ataque pode ser executado. É também possível acrescentar módulos de encapsulamento que representem outros protocolos.

Não foram gerados gráficos a partir desse algoritmo pois, a diferença entre os gráficos, do ataque *online* e do *offline*, seria imperceptível. Isso acontece por ser baixa a taxa de perda de pacotes, devido a ultrapassagem do tempo de *timeout*. Ou seja, o *trace* de ataque usado não é suficiente para gerar uma perturbação considerável no *trace* da máquina atacada.

6.3 Experimento de Ataque *Offline*

Com os traces obtidos do experimento de ataques *online*, em arquivos no formato pcap - *ataque_http_get_req_flood.pcap*, do ataque *HTTP GET Request Flood*; *ataque_icmp_echo_req.pcap*, do ataque *ICMP Echo Request Flood*; *ataque_ip_flood.pcap*, com o ataque *Protocol* 0 *Flood*; *ataque_tcp_syn_flood.pcap*, do ataque *TCP SYN Flood*; *ataque_udp80.pcap*, do ataque *UDP FLOOD*; e o *trace* normal - *base_normal.pcap* - conforme mencionado no último capítulo, foi feito um experimento *offline* que busca imitar um ataque de negação de serviço distribuído.

O objetivo do experimento é explorar como o tráfego de pacotes normal reagiria a múltiplos ataques. Para isso foi criado um aplicativo que intercala os cinco diferentes tipos de ataques, já mencionados acima, no *trace* do tráfego normal da rede.

6.3.1 Funcionamento do Aplicativo

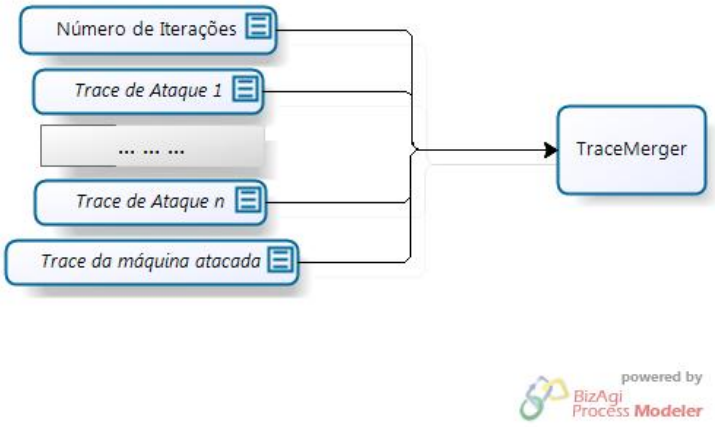


FIG. 6.6: Entradas do Aplicativo

A partir do arquivo *base_normal.pcap*, que representa um tráfego de rede convencional onde cada pacote que exige um reconhecimento - pacote *ACK* - o recebe, criam-se novos *traces* com a inserção de um ou mais dos cinco tipos de ataques de inundação de pacotes. Na prática, estas inserções são conseguidas com a intercalação, do verbo em inglês *to merge*, dos *traces* de ataque com o *trace* não atacado. O

processo de fundição é iterativo e seu ponto de parada é configurável. Um ciclo deste processo imita o que acontece na realidade - um bombardeamento de pacotes originados de uma única fonte à uma máquina alvo. O aplicativo funciona para um número qualquer de ciclos, com o único cuidado de não estourar o limite de memória de disco, e a ocorrência de múltiplos ciclos infere a característica do ataque ser distribuído.

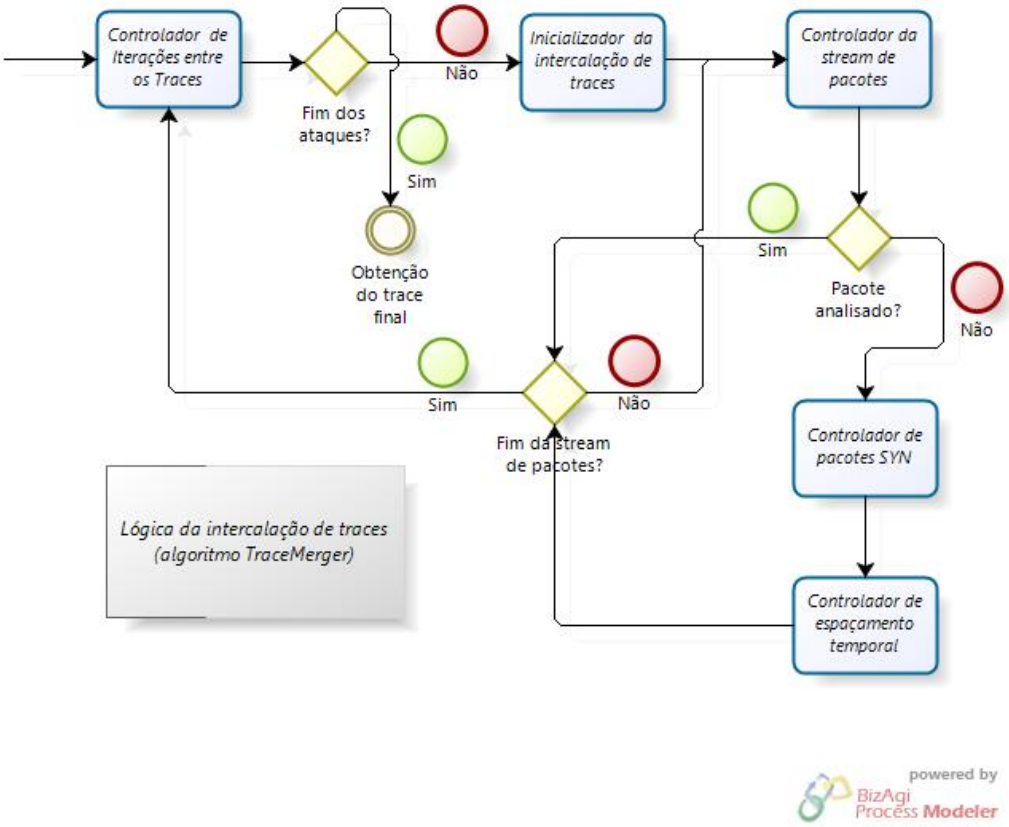


FIG. 6.7: Fluxo do Aplicativo

A FIG. 6.6 ilustra os argumentos de entrada do aplicativo, enquanto que a FIG. 6.7 detalha a arquitetura do aplicativo a partir dos seguintes componentes:

- Controlador de Iterações entre *Traces*: verifica o número de iterações restantes no algoritmo e seleciona o *trace* para o tipo de ataque escolhido randomicamente.
- Inicializador da Intercalação de *Traces*: inicia os leitores e escritores dos rastreamentos original e do ataque selecionado no Controlador de Iterações entre *Traces*.
- Controlador da *Stream* de Pacotes: responsável pelo controle de inicialização e término da stream de pacotes.
- Controlador de Pacotes *SYN*: monitora o *timeout* dos pacotes *SYN*.

- Controlador de Espaçamento Temporal: realiza operações de translação de tempo no *trace*. Modifica o campo time dos pacotes.

A lógica de negócio da aplicação se encontra na classe *TraceMerger.java*. Em seguida se encontra uma explicação do funcionamento da mesma.

6.3.2 Principais Campos

- *JpcapCaptor captorAtk*: captura e encapsula os pacotes de ataque utilizados para bombardeamento;
- *JpcapCaptor captorTarget*: captura e encapsula os pacotes da base original (máquina alvo);
- *JpcapWriter writer*: escreve sobre um arquivo *pcap*, ou seja, cria um *trace*;
- *Packet p1*: encapsula os pacotes das máquinas atacantes;
- *Packet p2*: encapsula os pacotes da máquina alvo;
- *long dt*: espaçamento de tempo, dado em milissegundos, entre a chegada de pacotes;
- *long shift_sec*: faixa temporal, dada em milissegundos, para se adicionar ao tempo de chegada entre pacotes - deslocamento de tempo;
- *long shift_usec*: faixa temporal, dada em microssegundos, para se adicionar ao tempo de chegada entre pacotes - deslocamento de tempo;
- *long timeout_tcp_syn*: faixa temporal, dada em milissegundos, para se adicionar ao pacote do tipo *SYN* que sofreu *timeout*.

6.3.3 Principais Métodos

- *void inicializar()*: é executado no início de cada iteração da rotina. Neste método, a máquina atacante escolhe qual técnica utilizará para bombardear a máquina alvo. Este método está implementado de maneira que a escolha ocorre de maneira randômica.
- *void shift()*: é executado para cada pacote de ataque que chega na placa de rede da máquina alvo o que ocorre várias vezes em cada iteração da rotina. Este método controla o espaçamento temporal, ou deslocamento, entre cada pacote na base atacada final. Despendendo do número de deslocamentos entre dois pacotes e do tipo dos mesmos, há a possibilidade de ocorrer um *timeout* e ocorrer perda de pacotes, daí, para cada pacote perdido, um novo pacote é enviado.
- *void executar()*: é a parte principal do aplicativo, onde se dão todas as iterações lógicas. O número de iterações é dado pelo usuário e é representado pela variá-

vel *numberOfIterations* e cada uma começa com a chamada ao método *inicializar()*. Com o *trace* de ataque selecionado, ocorre a inserção do mesmo na base original, ou seja, os *traces* são fundidos. Neste processo, o tempo de início do ataque é escolhido aleatoriamente entre o tempo do primeiro e do último pacote do rastreamento normal e os pacotes do mesmo são deslocados gradativamente pela chamada do método *shift()* para cada pacote inserido do *trace* atacante.

6.3.4 Encapsulamento do *Three Way Handshake* via *TCP/IP*

Uma das funcionalidades do aplicativo é simular o comportamento normal da rede, ou seja dos diversos protocolos de comunicação. Ao realizarmos um ataque *offline* usando a estratégia de *merge* corremos o risco de ferirmos algum protocolo, principalmente no que tange os conjuntos de *timeouts*. Para satisfazer esses protocolos se faz necessário então o encapsulamento de seu funcionamento. Neste trabalho, foi realizado apenas o encapsulamento da funcionalidade do *timeout* dos pacotes de início de conexão *TCP*.

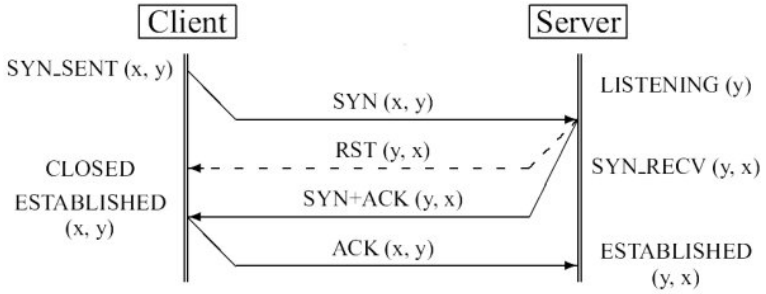


FIG. 6.8: Conexão 3 - *Way Handshake* (extraído de *www.usenix.org*)

A FIG. 6.8 ilustra o funcionamento do protocolo usado para iniciar-se uma conexão *TCP*. O encapsulamento desenvolvido nesse trabalho aborda a situação em que um pacote *SYN* expira o seu *timeout*. Observando a figura, o pacote *RST*, que contém a linha tracejada, representa que o servidor está ciente de que um pacote *SYN* ultrapassou seu limite de espera por uma resposta. O encapsulamento proposto irá então simular essa falha no início da conexão *TCP*. Essa escolha foi realizada tendo em vista que o ataque *TCP SYN Flood*, tem como característica principal atuar nessa falha de protocolo, e, também, por tal ataque ter sido executado e analisado no presente trabalho.

6.3.4.1 Implementação

Ao fazer o procedimento de "*merge*" entre os *traces* percorre-se cada pacote dos *traces* envolvidos na operação. A partir disso, os pacotes visitados que contenham a *flag SYN* ativada, porém, a *flag ACK* desativada são armazenados em um *array* redimensionável de objetos, através da classe *Vector*. Esse *array* vai permitir manipular, pontualmente, os pacotes de início de conexão *TCP* (*flag SYN* ativada e *flag ACK* desativada) que foram perdidos. Essa perda ocorre quando o *timeout* correspondente é ultrapassado.

Quando o pacote visitado possuir a *flag SYN* ativada e a *flag ACK* também ativada, esse pacote vai representar a resposta a algum pacote que foi previamente armazenado no *array*. Essa correspondência se dá através dos campos "número de sequência" e "número de *ACK*", onde o "número de *ACK*" do pacote de resposta é igual ao "número de sequência" do pacote de requisição acrescido de uma unidade. Sendo assim, o pacote correspondente armazenado no *array* é então removido do mesmo.

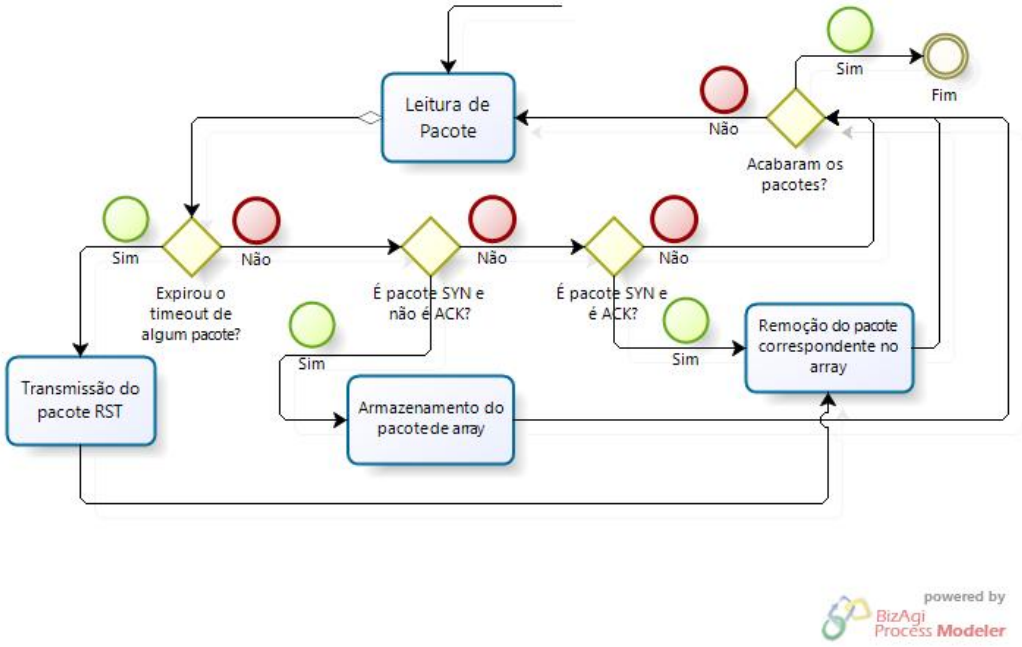


FIG. 6.9: Lógica do Encapsulamento

Em cada leitura de pacote são verificados todos os timeouts dos pacotes armazenados no *array*. Caso haja algum pacote que tenha ultrapassado o *timeout*, esse pacote é removido do *array* e um novo pacote é "transmitido" com a *flag RESET* ativada, demonstrando que o pacote precisa ser reenviado. A FIG. 6.9 ilustra o procedimento descrito.

Essa implementação não leva em conta todas as particularidades do protocolo de início e de encerramento de conexão. Isso ocorre por ser necessário um estudo mais detalhado do protocolo envolvido. Além disso, existe a problemática da complexidade da estrutura de dados que é necessária para representar todas as particularidades das regras desse protocolo.

7 CONCLUSÃO

É possível através do aplicativo implementado no presente trabalho gerar um *trace* de uma rede atacada por um conjunto de um ou mais ataques dos seguintes tipos: *icmp echo request flood*, *protocol 0 flood*, *tcp syn flood*, ataque *udp80 flood* ou *http get request flood*. A reconstrução do *trace* original com a inserção dos ataques supracitados visa retratar um ataque realista, de maneira que pacotes, do rastreamento original, após a inserção de ataques, são perdidos devido a falhas de *timeout* e são reenviados de acordo com as regras do protocolo de comunicação.

A seguir são apresentadas algumas sugestões para atividades futuras que complementariam o programa feito neste trabalho ou que podem vir a melhorar o desempenho da aplicação. São elas:

- Traduzir o programa para a linguagem *C*, utilizando a *API WinPcap*. Com isso, traces maiores poderão ser gerados com maior eficiência e o programa ficaria livre de alguns contratempos da *API Jpcap*;
- Com o item anterior, utilizar-se da inserção de rastreamentos de ataque mais extensos, com o objetivo de explorar o comportamento da máquina atacada para maiores tempos de *timeout*, ao compor o *trace* final;
- Implementar encapsulamentos do comportamento dos protocolos restantes, e, finalizar as etapas restantes do *three way handshake*.

8 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Ataque ao servidor do site *register.com*.
Disponível em: <http://staff.washington.edu/dittrich/misc/ddos/register.com-unisog.txt>. Acesso em Abril de 2010.

[2] Ataques DDoS.
Disponível em: <http://www.rnp.br/newsgen/0003/ddos.html>. Acesso em Abril de 2010.

[3] Twitter, Facebook attack targeted one user.
Disponível em: <http://news.cnet.com/8301-27080_3-10305200-245.html>. Acesso em Agosto de 2010.

[4] Ataques DoS e DDoS.
Disponível em: <http://www.infowester.com/col091004.php>. Acesso em Abril de 2010.

[5] Paul Virilio - Biography.
Disponível em: <http://www.egs.edu/faculty/paul-virilio>. Acesso em Agosto de 2010.

[6] ARQUILLA, John & RONFELDT, David. Cyberwar is Coming!
Disponível em: <http://www.rand.org/pubs/monograph_reports/MR880/MR880.ch2.pdf>. Acesso em Agosto de 2010.

[7] A Guerra de Quarta Geração, a Guerra em Rede Social e a Situação Atual em Honduras.
Disponível em: <http://www.coter.eb.mil.br/textos/2009%20-%20guerra%20em%20rede%20social.pdf>. Acesso em Agosto de 2010.

[8] Inside the Chinese Hack Attack.
Disponível em: <http://www.time.com/time/nation/article/0,8599,1098371,00.html>. Acesso em Agosto de 2010.

[9] Cyber War: Sabotaging the System.
Disponível em: <http://www.cbsnews.com/stories/2009/11/06/60minutes/main5555565.shtml>. Acesso em Abril de 2010.

[10] UK, Not North Korea, Source of DDOS Attacks, Researcher Says.
Disponível em: <http://www.pcworld.com/businesscenter/article/168353/uk_not_north_korea_source_of_ddos_attacks_researcher_says.html>. Acesso em Abril de 2010.

[11] A new approach to China.
Disponível em: <http://googleblog.blogspot.com/2010/01/new-approach-to-china.html>. Acesso em Abril de 2010.

[12] DoS - Negação de Serviço.
Disponível em: <http://www.gta.ufrj.br/grad/06_1/dos/intro.html>. Acesso em Abril de 2010.

[13] CERT Coordination Center, “Denial of Service Attacks”.
Disponível em: <http://www.cert.org/tech_tips/denial_of_service.html>. Acesso em Abril de 2010.

[14] Jelena Mirkovic, Janice Martin & Peter Reiher. *A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms.* Technical report #020018. Computer Science Department, University of California, Los Angeles

[15] J. Mirkovic & P. Reiher. *A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms.* ACM SIGCOMM Computer Communication Review, v.34 n.2, 2004

[16] Tripwire, “Tripwire for servers”.
Disponível em: <http://www.tripwire.com/products/servers/>. Acesso em Abril de 2010.

[17] S. Axelsson. *Intrusion detection systems: A survey and taxonomy.* Technical Report 99-15. Department of Computer Engineering, Chalmers University, 2000.

[18] C. Meadows. *A formal framework and evaluation method for network denial of service.* IEEE Computer Security Foundations Workshop, 1999.

[19] J. Leiwo, P. Nikander & T. Aura. *Towards network denial of service resistant protocols.* International Information Security Conference (IFIP/SEC 2000), 2000.

[20] A. Juels & J. Brainard. *Client puzzles: A cryptographic countermeasure against connection depletion attacks.* Networks and distributed system security symposium (NDSS’99), 1999.

[21] Y. L. Zheng & J. Leiwo. *A method to implement a denial of service protection base.* Information Security and Privacy, v.1270, LNCS, p.90–101, 1997.

[22] O. Spatscheck & L. Peterson. *Defending against denial-of-service requests in Scout.* USENIX/ACM Symposium on Operating System Design and Implementation, 1999.

[23] F. Lau, S. H. Rubin, M. H. Smith & Lj. Trajkovic. *Distributed denial of service attacks.* IEEE International Conference on Systems, Man, and Cybernetics, 2000.

[24] A. Garg & A. L. Narasimha Reddy. *Mitigating denial of service attacks using QoS regulation.* Texas A & M University Tech report, TAMU-ECE-2001-06.

[25] J. Yan, S. Early & R. Anderson. *The XenoService - A distributed defeat for distributed denial of service.* ISW 2000, 2000.

[26] S.Floyd, S. Bellovin, J. Ioannidis, K. Kompella, R. Mahajan & V. Paxson. *Pushback Messages for Controlling aggregates in the Network.* Trabalho em progresso, disponível em: <http://www.icir.org/floyd/papers/draftfloydpushbackmessages00.txt>. Acesso em Abril de 2010.

[27] Information Sciences Institute, “Dynabone”. *Disponível em: <http://www.isi.edu/dynabone/>.* Acesso em Abril de 2010.

[28] T. M. Gil & M. Poleto. *MULTOPS: a data-structure for bandwidth attack detection.* Usenix Security Symposium, 2001.

[29] Mananet, “Reverse Firewall”. *Disponível em: <http://www.cs3-inc.com/ps_rfw.html>.* Acesso em Abril de 2010.

[30] D. X. Song & A. Perrig. *Advanced and authenticated marking schemes for IP Traceback.* IEEE Infocom, 2001.

[31] D. Dean, M. Franklin & A. Stubblefield. *An algebraic approach to IP Traceback.* Network and Distributed System Security Symposium, 2001.

[32] S. M. Bellovin. *ICMP traceback messages.* *Trabalho em progresso, disponível em: <http://tools.ietf.org/id/draft-ietf-itrace-01.txt>.* Acesso em Abril de 2010.

[33] S. Savage, D. Wetherall, A. Karlin & T. Anderson. *Practical network support for IP Traceback.* ACM SIGCOMM Conference, 2000.

[34] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent & W. T. Strayer. *Hash-Based IP Traceback.* ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, 2001.

[35] P. Ferguson & D. Senie. *Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing.* RFC 2267, 1998.

[36] J. Li, J. Mirkovic, M. Wang, P. Reiher & L. Zhang. *SAVE: Source address validity enforcement protocol.* INFOCOM 2002, 2002.

[37] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek & R. Morris. *Resilient Overlay Networks.* ACM SOSP, 2001.

[38] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba & Kumar Das. *The 1999 DARPA Off-Line Intrusion Detection Evaluation.* Draft of paper submitted to Computer Networks, In Press, 2000.

[39] YEN, E. *Data mining-based intrusion detectors.* Expert Syst. Appl., Pergamon Press, Inc., Tarrytown, NY, USA, v.36, n.3, p.5605-5612, 2009.

[40] JUNIOR, Fábio Luiz de Sousa & BENEDITO, Vinícius Hessel. *Estudo de ataques distribuídos de negação de serviço.* Trabalho de Iniciação à Pesquisa, Instituto Militar de Engenharia, Rio de Janeiro, 2010.

[41] Jpcap. *Disponível em: <http://netresearch.ics.uci.edu/kfujii/jpcap/doc/>.* Acesso em Agosto de 2010.

[42] WinPcap: The Windows Packet Capture Library. *Disponível em: <http://www.winpcap.org/>.* Acesso em Agosto de 2010.