MINISTÉRIO DA DEFESA

EXÉRCITO BRASILEIRO

DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA

INSTITUTO MILITAR DE ENGENHARIA

(Real Academia de Artilharia, Fortificação e Desenho - 1792)

VINÍCIUS MAIA SENNA DELGADO

WALTER MACAMBIRA OLLIVEIRA SANTTOS

FERRAMENTA DE SIMULAÇÃO E OBTENÇÃO DE *TRACES* DE UM ATAQUE DDOS

Rio de Janeiro

2012

INSTITUTO MILITAR DE ENGENHARIA

VINÍCIUS MAIA SENNA DELGADO

WALTER MACAMBIRA OLLIVEIRA SANTTOS

FERRAMENTA DE SIMULAÇÃO E OBTENÇÃO DE *TRACES* DE UM ATAQUE DDOS

Projeto de Final de Curso apresentado ao Curso de Graduação de Engenharia de Computação como requisito parcial para a obtenção do título de Engenheiro.

Orientadores: Anderson Fernandes Pereira dos Santos (D. Sc.) e Sérgio dos Santos Cardoso Silva (M. Sc.).

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

621.39 Santtos, Walter Macambira Olliveira; Delgado, Vinícius Maia Senna.

D352f Ferramenta de Simulação e Obtenção de *Traces* de um Ataque *DDoS*. Santtos, Walter Macambira Olliveira; Delgado, Vinícius Maia Senna; orientado por Anderson Fernandes Pereira dos Santos, Sérgio dos Santos Cardoso Silva – Rio de Janeiro: Instituto Militar de Engenharia, 2012.

75f : il.

Projeto de Final de Curso – Instituto Militar de Engenharia – Rio de Janeiro, 2012.

1. Engenharia de Computação 2. Negação de Serviço – ataque 3. Negação de Serviço – ferramenta 4. Segurança da Informação I. Santtos, Walter Macambira Olliveira II. Santos, Anderson Pereira Fernandes III. Delgado, Vinícius Maia Senna IV. Silva, Sérgio dos Santos Cardoso V. Título VI. Instituto Militar de Engenharia

CDD 621.39

INSTITUTO MILITAR DE ENGENHARIA

VINÍCIUS MAIA SENNA DELGADO

WALTER MACAMBIRA OLLIVEIRA SANTTOS

FERRAMENTA DE SIMULAÇÃO E OBTENÇÃO DE *TRACES* DE UM ATAQUE DDOS

Relatório final do Projeto de Final de Curso apresentado ao Curso de Graduação de Engenharia de Computação como requisito parcial para a obtenção do título de Engenheiro.
Orientadores: Anderson Fernandes Pereira dos Santos – D. Sc.
Sérgio dos Santos Cardoso Silva – M. Sc.

Aprovado em 25 de maio de 2012 pela seguinte Banca Examinadora:

Anderson Fernandes Pereira dos Santos – D. Sc.

Sérgio dos Santos Cardoso Silva – M. Sc.

Julio Cesar Duarte – D. Sc.

Raquel Coelho Gomes Pinto – D. Sc.

Rio de Janeiro

2012

AGRADECIMENTOS

Dedicamos o referido trabalho às nossas famílias, que sempre estiveram presentes em nossa formação e nos incentivaram a prosseguir em nossos projetos. Aos nossos colegas de turma, com os quais não compartilhamos somente alegrias, mas dificuldades das quais nunca nos escusamos. Especialmente, aos nossos amigos Wagner e Valesca pelo apoio, conselhos e incentivo nas horas de estudo. Ao Maj Duarte e à professora Raquel, pela contribuição com o texto. E, principalmente, ao Maj Anderson e ao Maj Cardoso, nossos orientadores, pelo constante entusiasmo com o projeto e pela prontidão em sempre nos auxiliar no curso desta pesquisa.

Sumário

LISTA DE SIGLAS

API	*Application Programming Interface*
AS	Sistema Autônomo
ASCII	*American Standard Code for Information*
BA	*Barabási-Albert*
BFS	*Breadth First Search*
BGP	*Border Gateway Protocol*
BGP	*Border Gateway Protocol*
BRIANA	*BRITE Analysis Engine*
BRITE	*Boston University Representative Internet Topology Generator*
BS	*British Standard*
CAIDA	*Cooperative Association of Internet Data Analysis*
DDoS	*Distributed Denial of Service*
DNS	*Domain Name System*
DoS	*Denial of Service*
Gbps	Gigabits por segundo
GloMoSim	*Global Mobile Information System Simulator*
GLP	*Generalized Linear Preference*
GNU GPL	*GNU General Public License*
GT-ITM	*Georgia Tech Internetwork Topology Model*
GUI	*Graphical User Interface*
HTTP	*Hypertext Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IEC	*International Electrotechnical Comission*
IME	Instituto Militar de Engenharia
IMUNES	*Integrated Multiprotocol Network Emulator/Simulator*
INRIA	*Inventeurs Du Monde Numérique*
IP	*Internet Protocol*
ISP	*Internet Service Provider*
ISSO	*International Organizational for Standardization*
LAN	*Local Area Network*
LOIC	*Low-orbit Ion Cannon*
LRD	*Long-range Dependece*
MAN	*Metropolitan Area Network*

MAWI	*Measurement and Analysis on the WIDE Internet*
Mbps	Megabits por segundo
NCTUns	*National Chiao Tung University Network Simulator*
NED	*Network Description*
NLANR	*Natriona Laobratory for Applied Networking Research*
NS-3	*Network Simulator 3*
OPNET	*Optimized Network Engineering Tools*
OSPF	*Open Shortest Path First*
OTcl	*Object Tool Command Language*
PCAP	*Packet Capture*
PPBP	*Poisson Pareto Burst Process*
SIP	*Session Initiation Protocol*
SSQ	*Single Server Queuee*
TCP	*Transfer Control Protocol*
UCLA	*University of California – Loas Angeles*
UDP	*User Datagram Protocol*
VoIP	*Voice over IP*
WAN	*Wide Area Network*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A internet é um meio de comunicação cada vez mais valioso para a humanidade. Através da rede mundial de computadores, trafegam informações como segredos industriais, transações bancárias, documentos sigilosos e muitas outras. A disponibilidade dos meios para concretização dessas trocas de informação é, portanto, de extrema valia.

Um ataque de negação de serviço visa a indisponibilizar um serviço, máquina ou rede alvo. Essa indisponibilidade pode causar sérios prejuízos ao alvo do ataque. Geralmente, ataques de negação de serviço a determinado alvo procuram esgotar os seus recursos ou torná-lo inacessível para usuários legítimos.

O presente trabalho tem por finalidade desenvolver uma ferramenta capaz de simular um ataque distribuído de negação de serviço. Seu propósito é obter o máximo de informações possíveis para auxiliar o desenvolvimento de mecanismos de defesa. O desafio de realizar essa simulação encontra-se na complexidade para modelar principalmente a topologia e o tráfego de fundo, além de outros aspectos.

ABSTRACT

The Internet has become a valuable medium to mankind. Through the world wide web travels information such as industry secrets, banking transactions, secret documents and many others. The availability of the means to achieve this exchanges of information is of great importance.

A denial of service aims to cripple a service, a target machine or a network. The unavailability can cause serious damage to the target of the attack. Generally, denial of service attacks try to exhaust the resources of the target or make it inaccessible to legitimate users.

This work aims to develop a tool to simulate a distributed denial of service to gather information to assist the development of defense mechanisms. The challenge of performing these simulations is at the complexity of topology and traffic modeling as close as possible to reality, besides other aspects.

1. Introdução

O ataque de negação de serviço é potencialmente perigoso para o funcionamento de uma rede. Os danos imediatos são a interrupção do serviço ou a indisponibilidade de um recurso computacional. Os danos mais graves podem incluir exploração de vulnerabilidades decorrentes do DDoS (*Distributed Denial of Service*). Por esse motivo, realizar testes na rede para verificar a resistência de determinado serviço a esse tipo de ataque é inviável.

O caminho natural para o teste, desenvolvimento de ferramentas e técnicas de defesa para ataques DDoS é a simulação. Através desse artifício, é possível realizar testes de maneira segura e confiável. Como a simulação procura reproduzir fielmente o ambiente a ser simulado, em redes de computadores, pode-se obter esse resultado virtualmente (por meio de *software*) ou fisicamente.

Reproduzir fisicamente a totalidade ou fração de uma rede para a simulação aumenta a fidelidade e a confiabilidade dos resultados, porém essa solução pode ser economicamente inviável. Além dos ajustes e compras de novos equipamentos exigidos pela variação de parâmetros da simulação desejada, esses testes empregam uma escala que, normalmente, excede milhares de nós. Desta forma, o custo da simulação não compensa.

Assim, a simulação em ambiente virtual é uma solução mais adequada, mas deve ser realizada com critério para garantir a proximidade de seus resultados com o comportamento real de uma rede. Esses comportamentos devem ser modelados e validados de forma a garantir a confiabilidade dos resultados obtidos. Dois grandes desafios dessa modelagem são gerar uma topologia suficientemente representativa da Internet e reproduzir o seu tráfego de fundo, visto que estes assuntos ainda são motivo de constante pesquisa na comunidade científica.

Esse projeto faz parte de um esforço do Instituto Militar de Engenharia em produzir uma base de conhecimento sobre DDoS. Nessa instituição, estão sendo realizadas pesquisas sobre diversos assuntos relacionados ao tema. Desta forma, o trabalho anterior Implementação de um Ataque de Negação de Serviço no Ambiente

Network Simulator (SOUZA e JÚNIOR, 2011) foi utilizado como ponto de partida para o presente projeto. Os seus resultados foram reproduzidos e estão expostos no Anexo B.

No capítulo dois, será apresentada uma revisão bibliográfica sobre o ataque distribuído de negação de serviço. O terceiro capítulo trata da escolha do simulador a ser usado no trabalho, o quarto esclarece como o gerador de topologia foi escolhido. O quinto capítulo explica a teoria de modelagem do tráfego de fundo, o sexto aborda a construção da ferramenta utilizando os conceitos dos capítulos anteriores. O sétimo capítulo descreve uma instância de simulação e, finalmente, o oitavo capítulo traz os resultados e conclusões.

1.1. Objetivos

O objetivo geral do trabalho é produzir um componente de *software* que simule e registre informações trafegadas por uma rede durante um ataque de negação de serviço. Esse registro é comumente chamado de *trace* e será referido como tal nesse trabalho.

Para o desenvolvimento dessa ferramenta, os objetivos intermediários são:

- Relacionar os aspectos envolvidos na simulação de um ataque distribuído de negação de serviço.
- Realizar uma revisão bibliográfica sobre esses aspectos relacionados à simulação desejada.
- Estudar e configurar as ferramentas envolvidas no projeto.
- Simular um ataque distribuído de negação de serviço.

2. Ataques Distribuídos de Negação de Serviço

O projeto para se criar a Internet, levando em conta a sua escalabilidade e abertura, desconsiderou a segurança. Dada a facilidade para gerar tráfego e a falta de recursos que possibilitem a verificação da validade destes pacotes gerados, discernir qual parte do tráfego é legítima se tornou o principal desafio. Essa falta de segurança possibilitou a existência de uma classe de problemas conhecidos como ataques de negação de serviço (DoS – *Denial of Service*).

A literatura define um ataque de negação de serviço como uma tentativa de negar acesso a usuários legítimos de recursos ou serviços compartilhados, com um contexto que varia de sistemas operacionais a serviços de rede (GLIGOR, 1984). Existem duas formas conhecidas para a realização desse tipo de ataque: a criação específica de pacotes para explorar vulnerabilidades de sistemas e a sobrecarga dos meios computacionais com tráfego não legítimo para ocupar todos os recursos disponíveis a usuários legítimos. Como a primeira forma pode ser evitada através de novas versões de *software*, o presente trabalho se concentrará na segunda, cuja dificuldade de prevenção é significativa. Como fator complicador, podem-se empregar diversas fontes para a realização desse tipo de ataque, transformando-o em um ataque distribuído de negação de serviço (DDoS – *Distributed Denial of Service*).

Qualquer serviço que se baseie na Internet está sujeito a esses ataques, desde um simples servidor web a complexos sistemas de transações bancárias ou de emergência. Como atualmente se tornou conveniente migrar quase todos os serviços convencionais para aplicações desse tipo, é importante estudar a maneira correta de mitigar os danos causados por um ataque DDoS.

2.1 Segurança da Informação

Segurança da informação é a proteção do transporte e armazenamento de informações essenciais a uma organização. Seu objetivo é assegurar a confidencialidade, integridade, disponibilidade e autenticidade dos dados (ISO/IEC 27000).

A evolução da tecnologia da informação em suas diversas mídias requer cuidados para que indivíduos e organizações consigam armazenar e transitar informações de modo seguro. Desde os primórdios do transporte e armazenamento de informação em meios físicos, é possível notar métodos para mantê-la segura. Os romanos, por exemplo, já possuíam esquemas criptográficos para troca de mensagens.

A segurança da informação está bem definida nas seguintes normas internacionais BS 7799 (*British Standard*) e ISO/IEC 27000 (Organização Internacional para Padronização).

2.2 Segurança de Rede

Segurança de rede é toda atividade voltada à proteção de uma rede de informações. A segurança de rede deve prover cinco quesitos à informação trafegada, esses quesitos são divididos nas seguintes categorias conforme a Figura 2.1:

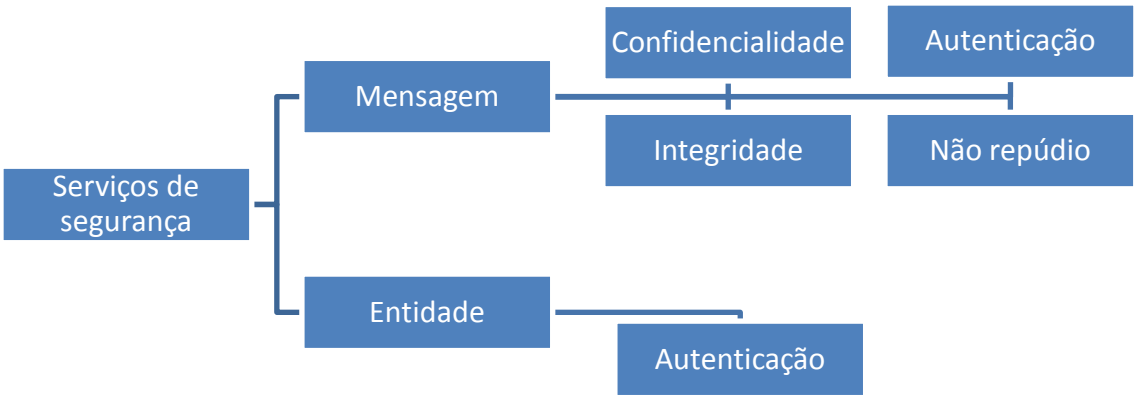

Figura 2.1 - Serviços de segurança de rede (FOURUZAN, 2006)

Confidencialidade é a capacidade de fazer com que a informação trafegada em uma rede só faça sentido ao emissor e ao receptor. Outros elementos da rede não devem tomar conhecimento da informação.

Integridade é a garantia de que a mensagem trafegada na rede não contenha erros, ou seja, a informação no emissor é exatamente igual no receptor.

Autenticação da mensagem diz respeito à capacidade de se assegurar a verdadeira autoria da mensagem recebida evitando que impostores participem da comunicação.

Não repúdio significa que o emissor não deve ser capaz de negar o envio de determinada mensagem que de fato ele enviou.

Autenticação de entidade assegura que somente os indivíduos desejados tenham acesso à rede.

2.3 Ataques DoS e DDoS

Um ataque de negação de serviço (DoS) normalmente caracteriza-se pelo envio de um grande volume de pacotes para ocupar parte significativa da banda disponível. Para que o serviço seja negado a usuários legítimos, o ataque em questão procura consumir recursos críticos de um serviço de rede, como, por exemplo, a capacidade de processamento de um servidor ou o espaço na pilha em um *software* de protocolo de rede.

Como o que importa é o volume de dados e não o conteúdo de cada pacote, o tráfego gerado em um ataque pode se assemelhar em conteúdo ao tráfego de usuários legítimos, dificultando significativamente a defesa do sistema alvo.

Antigamente, a infraestrutura para acesso à Internet oferecia pequena disponibilidade de largura de banda. Desta forma, para efetuar um ataque DoS, o realizador do ataque conseguia acesso a computadores com largura de banda superior ao alvo, possibilitando o ataque.

Com os avanços tecnológicos, o acesso à banda larga se popularizou e a antiga abordagem empregada para realizar ataques DoS se tornou praticamente ineficiente. Além disto, as máquinas se tornaram mais potentes e diversas falhas de *software* foram corrigidas. Era necessário aumentar o volume de dados para o sucesso de um ataque. Como solução, passou-se a utilizar diversas fontes para gerar esse tráfego para a vítima, caracterizando o que se chama de ataque distribuído de negação de serviço.

Para realizar um ataque DDoS, deve-se inicialmente comprometer sistemas vulneráveis com acesso à Internet por meio de *malwares*. Esse tipo de programa abre caminho para a instalação de ferramentas de ataque. Estes computadores infectados são chamados de zumbis ou *bot*. Em seguida, o realizador do ataque, através de um canal de comunicação qualquer, ordena que esses zumbis iniciem de maneira sincronizada o ataque contra uma ou mais vítimas (DIETRICH, 2000).

Naturalmente, o número de zumbis pode variar de dezenas à ordem de milhões (MESSMER, 2009). Como o protocolo IP permite a alteração do endereço de origem, a detecção de que um ataque DDoS está em curso é prejudicada. Além disto, o tráfego gerado pode superar o limite da maioria dos *links* coorporativos (ARBOR, 2005) e exceder o *throughput* da maioria dos dispositivos de segurança de rede. Devido à distribuição geográfica que o ataque pode assumir, torna-se praticamente impossível realizar o *traceback* dos endereços de origem. Caso a rede de zumbis seja muito grande, o tráfego individual de cada maquina poderá ser reduzido, dificultando ainda mais sua diferenciação do trafego legitimo

Com a popularização da banda larga e a falta de conhecimento dos usuários, a infecção de computadores para torná-los zumbis se tornou uma tarefa relativamente simples. Através de ataques diretos, é possível explorar vulnerabilidades de *softwares* que oferecem serviços públicos nos computadores-alvo, transformando-os em zumbis. Vale ressaltar que este tipo de ataque pode ser automatizado. Em contrapartida, existem os ataques indiretos, que exigem que o usuário execute uma determinada ação que possibilita a infecção. Um conjunto de *bots* coordenados sob determinado canal de comunicação forma uma rede denominada *botnet*.

Atualmente, com a crescente visibilidade dos ataques distribuídos de negação de serviço como forma de ativismo político, os realizadores dos ataques investiram no desenvolvimento de ferramentas, como a LOIC (*Low Orbit Ion Cannon*), com as quais qualquer pessoa pode se voluntariar para se tornar um zumbi. Essa nova abordagem, conhecida como “hacktivismo”, possibilitou a formação de *botnets* com proporções ameaçadoras para a segurança dos sistemas de informação (MANSFIELD, 2011).

2.4 Técnicas de Ataque

A força de um ataque pode ser definida em termos da quantidade de recursos da vítima consumidos por consequência do ataque. Dos fatores contribuintes para essa força, destacam-se normalmente o volume de tráfego em um período e a quantidade de recursos que cada pacote consome.

Ataques de negação de serviço causam o consumo dos recursos computacionais da vítima, fazendo com que a mesma comece a não ler alguns dos pacotes recebidos devido à falta de recursos. Nos usuários legítimos, essas perdas ativam mecanismos que causam redução da taxa de transmissão de pacotes. Desta maneira, os zumbis ganham espaço para aumentar seus tráfegos. Com o tempo, os recursos da vítima se esgotam, tornando-se impossível servir usuários legítimos.

Além disto, esses ataques consomem a largura de banda da rede, ou seja, é possível que os fluxos gerados pelos zumbis dominem os *links* de comunicação, bloqueando qualquer fluxo legítimo. Deste modo, além da vítima não conseguir prover o serviço, esses *links* se tornam inutilizáveis por quaisquer outros serviços.

Apesar da classificação para os ataques não ser mutuamente exclusiva, pode-se categorizá-los em: os que tiram proveito dos protocolos, os que têm uma aplicação específica como alvo, os que utilizam terceiros para amplificar o ataque e os que derrubam a infraestrutura da Internet.

2.4.1 Ataques Baseados no Protocolo

Esse tipo de ataque não exige um grande número de zumbis para sua realização, podendo, às vezes, ser realizado a partir de uma única máquina. Seu poder de ataque se deve a fraquezas específicas do protocolo empregado.

Como exemplo, o ataque *SYN Flood* tira proveito do *three-way handshake* previsto pelo protocolo TCP, o cliente da conexão deixa de enviar uma confirmação tornando a conexão ilegítima. Já o ataque do tipo *Smurf* faz proveito do protocolo

ICMP, confiando a sobrecarga da vítima nas respostas das requisições de *echo* normalmente propagadas pela maioria dos *hosts* de uma rede.

2.4.2 Ataques Baseados na Aplicação

A camada de aplicação não possui um padrão de segurança normatizado e, comumente, exige um elevado aumento de custo para sua implementação. Por isso, ela se mostra mais susceptível a vulnerabilidades, inclusive aos ataques de largura de banda. Desta forma, o realizador do ataque solicita a execução de tarefas que consomem muitos recursos, causando o esgotamento dos recursos da vítima.

Para exemplificar esse tipo de ataque, pode-se citar o *HTTP Flood* e o *SIP Flood*. O primeiro se baseia na simulação do tráfego legítimo através de solicitações HTTP. Já o segundo tipo se deve à popularização do VoIP, aproveitando-se do funcionamento dos pacotes de convite que o protocolo SIP provê. Este último tipo de ataque afeta tanto os recursos do servidores *proxy* SIP como a capacidade da rede.

2.4.3 Ataques com Refletores Distribuídos

Nesse tipo de ataque, os zumbis não enviam os pacotes diretamente para a vítima, mas sim para terceiros inocentes, chamados de refletores, como se fossem a vítima. Em seguida, esses terceiros responderão os pacotes recebidos à vítima.

O emprego de refletores resolveu um dos grandes problemas dos ataques: o sigilo do endereço dos zumbis. Essa dispersão causada dificulta ainda mais a detecção do ataque e a descoberta de sua origem. Além disto, o uso de refletores amplifica o tráfego do ataque (GIBSON, 2002), como será mostrado a seguir.

O DNS (*Domain Name System*) é uma infraestrutura distribuída para associar domínios à registros de recursos (RR). De maneira simplificada, serve para traduzir domínios em endereços IP. Para resolver uma requisição, ela é inicialmente feita a um servidor raiz, que retornará o endereço do servidor de topo responsável (*top-level*

domain). Este, por sua vez, retornará o endereço do servidor autoritativo, que contém a informação, ou de um intermediário. Neste segundo caso, será retornado o endereço de outro servidor. Caso essa requisição seja resolvida sem interação com o cliente nos passos intermediários, trata-se de um servidor recursivo.

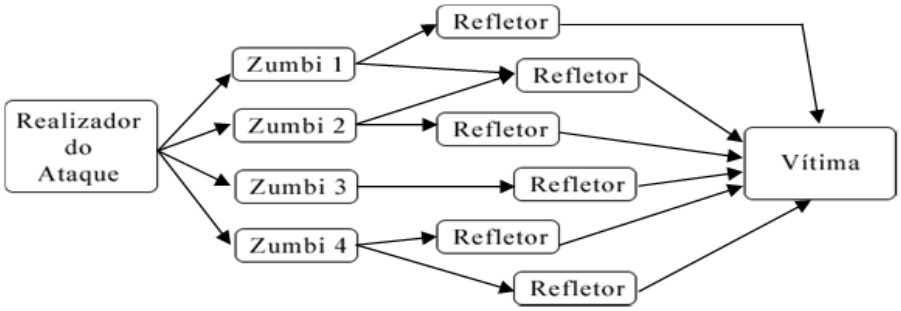

Figura 2.2 – Esquema de um ataque com refletores distribuídos

Um ataque de amplificação de DNS se inicia com o comprometimento de um servidor DNS autoritativo. Nele, inserem-se registros de texto com tamanho de alguns *kilobytes*. Na verdade, esta fase é opcional, visto que já existem registros públicos legítimos com tamanhos semelhantes. Em seguida, o realizador do ataque solicita esses registros forjados a vários servidores de DNS recursivos, que fazem a busca e armazenam a resposta em *cache*. O cenário está preparado para a realização do ataque propriamente dito. Para isso, ordena-se que uma *botnet* faça requisições DNS do registro inserido, estabelecendo como endereço de origem o IP da vítima do ataque. Como os servidores consultados são recursivos, eles apenas interagem com a vítima para retornar o registro encontrado. A amplificação gerada pode tornar um tráfico inicial de 140Mb/s da *botnet* em 10Gb/s para a vítima (SCALZO, 2006). A Figura 2.2 ilustra este tipo de ataque.

2.4.4 Ataques à Infraestrutura

O objetivo desse tipo de ataque é desabilitar serviços de componentes críticos da Internet, sendo potencialmente catastrófico para o funcionamento de, até mesmo, toda a rede mundial de computadores. Obviamente, esses componentes são produzidos para conseguir lidar com quantidades elevadíssimas de tráfego.

2.5 Técnicas de Defesa

Para se defender de um ataque distribuído ou não de negação de serviço, existem quatro estágios principais: prevenção, detecção, identificação de fonte e reação. A defesa se inicia com a prevenção e termina com a reação. Nenhuma dessas quatro etapas isoladamente se configura como uma solução ideal, sendo a integração de todas necessária para mitigar os riscos de um ataque distribuído de negação de serviço.

Na primeira etapa, tenta-se prevenir os danos do ataque com a filtragem de pacotes com endereços de origem falsos, ou seja, evitar que os pacotes cheguem ao seu destino. Como o próprio nome diz, a técnica de detecção procura detectar a ocorrência do ataque para a tomada de decisões. A terceira técnica tenta identificar o remetente do pacote independente da veracidade a respeito de seu endereço informado. A última das quatro etapas, a reação, busca reduzir ou eliminar os efeitos de um ataque, encontrando dificuldade em não afetar o tráfego legítimo. Ela é o último recurso dos estágios de proteção de um sistema, sendo o fator determinante para o desempenho do mecanismo de defesa.

2.5.1 Prevenção de Ataque

Nesse contexto, o termo prevenção é empregado com o sentido de evitar os danos do ataque. Isso é feito por meio de filtros capazes de impossibilitar a chegada dos pacotes aos seus destinos. Vários esquemas de filtragem de pacotes são configurados nos roteadores, permitindo a passagem apenas de trafego legítimo. De toda forma, especificar as regras para filtragem não é uma tarefa tão simples, além de exigir uma grande difusão para se tornarem efetivas. Naturalmente, este tipo de defesa funciona somente contra técnicas de ataque que informam endereços de remetente falsos.

2.5.2 Detecção de Ataque

Um esquema de detecção de ataques de negação de serviço se guia pela perda de desempenho dos serviços da vítima, preocupando-se com os recorrentes falso-

positivos[1], dado que um volume maior de tráfego mesmo que legítimo ("*flash crowds*") pode ser encarado como um ataque. Detectar um ataque antes de sua ocorrência possibilita que a vítima consiga reagir adequadamente, protegendo seus usuários legítimos. Além disto, a detecção auxilia na identificação dos responsáveis pelos ataques e ainda, caso ocorra perto da fonte,filtra o tráfego antes que seja desperdiçada banda. Naturalmente, não se trata de uma tarefa simples, visto que um ataque DoS não gera tanto tráfego perto da fonte em seu início.

A taxa de falso-positivos e o tempo de detecção devem ser baixos para que o esquema adotado seja eficiente. As técnicas para esse tipo de defesa podem se basear na detecção de anomalias ou de um tipo específico de ataque de negação de serviço.

2.5.3 Identificação de Fonte

Após detectado um ataque, seria adequado bloquear qualquer tráfego diretamente na fonte. Como o protocolo IP possibilita que facilmente sejam forjados endereços de origem e como o roteamento IP é *stateless* (sem estado), detectar a fonte de determinado tráfego é uma tarefa árdua.

Para resolver essa falta de rastreabilidade do protocolo IP, vários esquemas baseados em funções de roteamento melhoradas e em modificações dos protocolos atuais foram sugeridos. Dentre esses, pode-se citar os seguintes tipos de *traceback* de IP: por métodos probabilísticos e por base em *hashes*.

2.5.4 Reação ao Ataque

Todas as técnicas de defesa abordadas anteriormente buscam prover uma maneira de detectar um ataque e informar suas fontes no menor intervalo de tempo

[1] O tráfego incomum para uma determinada rede ou serviço não está necessariamente associado à ocorrência de um ataque distribuído de negação de serviço, podendo causar o que se chama falso-positivo. Como exemplo, pode-se citar o súbito crescimento de acessos à página do IME na data de divulgação do resultado do vestibular.

possível. Esta preocupação se deve à natureza agressiva e explícita de um ataque de negação de serviço. Desta forma, procura-se minimizar os danos causados pelo ataque de negação de serviço, definindo um esquema de reação a ser empregado quando o ataque estiver em curso.

Todos os sistemas possuem gargalos que, se comprometidos, impossibilitariam o restabelecimento do serviço. Desta forma, a reação inicial a ser tomada é a proteção destes gargalos, sendo denominada de *bottleneck resource management*.

A não ser que o tráfego seja filtrado na origem, recursos serão desperdiçados, diminuindo a qualidade do serviço para qualquer *host*, incluindo o do próprio alvo. É importante ressaltar que, após a reação inicial, outras reações devem ser tomadas em seguida antes que o grande volume de tráfego consiga criar novos gargalos.

Quando a reação é tomada em roteadores que se encontram entre o realizador do ataque e a vítima, ela é denominada de reação na rede intermediária. Filtrar o tráfego diretamente na fonte, que é considerada a solução ideal, se denomina de reação no terminal de origem.

2.6 Botnets

Dos computadores conectados à Internet, 16 a 25% deles fazem parte de uma *botnet* (ASSADHAN, 2009). Existem poucos estudos que confirmem esses dados e que descrevam as topologias decorrentes da expansão dessas redes. Desta forma, torna-se difícil a simulação da localidade dos nós infectados.

A cada quatro meses, a empresa Prolexic, especializada em ataques DDoS, publica um relatório que contém dados sobre a origem dos ataques DDoS por país. Essa proporção, entretanto, não fornece muitas informações sobre a localidade dos nós de uma única *botnet*, pois ela pode estender-se através da Internet sem levar em conta fronteiras geográficas (PROLEXIC, 2012). A localização dos *bots* de uma mesma subrede é resultado de seu mecanismo de expansão, tornando seu mapeamento complexo.

O mesmo relatório fornece informações sobre o tempo médio dos ataques e a banda média dos mesmos. No primeiro quadrimestre de 2012, as médias foram de 28,5 horas e de 6,1 Gbps. O tipo de ataque mais comum foi o *SYN Flood* (24%) seguido pelo *GET Flood* (20%), *ICMP Flood* (19%) e *UDP Flood* (15%). No último quarto de 2011, os mesmo quatro tipos de ataque foram os mais frequentes, porém na seguinte ordem: *UDP Flood* (22%), *SYN Flood* (21%), *GET Flood* (20%) e *ICMP Flood* (17%).

3. Simuladores de Rede

A implementação e projeto de um simulador de eventos discretos para a simulação de redes não é trivial. É preciso modelar todas as funcionalidades de simulação e ainda introduzir os conceitos de rede, seus protocolos e aplicações. A saída para esse problema é usar uma solução de software pronta e que atenda as necessidades desse trabalho.

Um simulador para um ataque de negação de serviço deve ser capaz de prover modelos confiáveis para os aspectos críticos do DDoS, sendo eles:

- Capacidade de processamento dos dispositivos da rede;
- Disponibilidade de enlaces e dispositivos;
- Largura de banda dos meios de comunicação;
- Capacidade de memória dos nós (*buffers*); e
- Tráfego de fundo para a rede representada.

Atualmente, nenhum simulador disponível no mercado oferece modelos satisfatórios para todos os aspectos. Particularmente, a capacidade de processamento dos nós da rede e a geração de tráfego de fundo são itens que, normalmente, os simuladores não implementam por serem de difícil representação.

Além desses aspectos críticos, o simulador deve suportar todos os aspectos básicos de uma rede, como a topologia, o comportamento dos nós segundo os protocolos e a comunicação entre os nós.

Alguns projetos do IME já trabalham com o NS-3, no entanto, é necessário reavaliar as opções existentes para verificar se uma solução mais adequada está disponível.

Os simuladores pesquisados para a realização desse trabalho foram os seguintes:

- GloMoSim;
- OPNET Modeler;
- OMNeT++
- NCTUns;

- GTNetS
- IMUNES; e
- NS-3.

Existem diversos simuladores de rede disponíveis no mercado. Algumas soluções foram descartadas por não se enquadrarem nos requisitos para esse trabalho, principalmente projetos muito antigos. Todos os programas que funcionavam exclusivamente como emuladores de rede também foram descartados, pois o propósito de um emulador é representar uma rede real de computadores funcionando em tempo real ao passo que a simulação utiliza modelos de abstração (Boeing Company, 2012).

Nenhuma das ferramentas analisadas apresentou diferencial que justificasse a adoção de uma solução diferente da adotada nas pesquisas do IME. Assim, o projeto seguirá a atual linha de pesquisa, adotando o NS-3. De toda forma, serão descritas as outras soluções avaliadas. Posteriormente, será feita a análise mais detalhada do NS-3.

3.1. Descrição dos simuladores avaliados

3.1.1. GloMoSim

O GloMoSim foi desenvolvido pela UCLA (*University of California - Los Angeles*) no final da década de 90 e tem como principal objetivo simular redes de dispositivos móveis. Esse simulador foi desenvolvido sobre outro simulador da UCLA, o Parsec, esse por sua vez é voltado a sistemas complexos, feito para a execução sequencial e em paralelo de simulações com eventos discretos.

Ambos foram escritos em linguagem C e tem como plataformas: Windows NT/2000, Linux RedHat 6+, Solaris 2.5.1+, entre outros. O GloMoSim foi inspirado no modelo OSI visando a integrar diversos modelos implementados por diferentes desenvolvedores.

As grandes desvantagens desse simulador são:

- Sua desatualização do código e das plataformas suportadas;

- Escassez de artigos científicos relacionados ao GloMoSim.

3.1.2. OPNET Modeler

O OPNET Modeler é um simulador proprietário que implementa em torno de oitocentos protocolos e aplicações de rede. Possui interface gráfica, suporte para simulação distribuída e diversas funcionalidades.

A ferramenta é paga, porém ela oferece uma versão gratuita para uso acadêmico. Essa versão tem o código aberto e é possível realizar a integração do simulador com outros programas em C++. A empresa oferece suporte técnico por e-mail para um professor e mais dois indivíduos cadastrados por universidade.

O manual do OPNET Modeler não está disponível em sua página virtual e, por isso, não foi possível realizar um estudo mais aprofundado sobre este simulador. Além disso, a empresa requer um cadastro para envio do simulador o que pode gerar atrasos para o projeto.

3.1.3. OMNeT++ (*Objective Modular Network Testbed in C++*)

O OMNet++ é uma ferramenta de simulação de eventos discretos de redes de computadores, multiprocessadores e outros sistemas distribuídos. É possível estendê-lo para simulação de outros sistemas através da adição de módulos.

O programa foi desenvolvido de forma a ser bastante modularizado e para possuir independência entre o controle da simulação e a abstração da simulação. Por esse motivo, existem diversas implementações de modelos da Internet dificultando a escolha da mais adequada.

O projeto dessa solução especifica uma linguagem de definição de topologia NED (*Network Description*). Não é prevista nenhuma integração com geradores de topologia e alguns de seus modelos já possuem a topologia fixada.

3.1.4. NCTUns

O NCTUns foi desenvolvido primeiramente em meio acadêmico e posteriormente comercializado com o nome de EstiNet. Uma grande vantagem do NCTUns é a utilização da pilha de protocolos do *kernel* do Linux para gerar sua simulação. Essa característica permite que o simulador também funcione como um emulador de redes.

O simulador foi desenvolvido e testado para o Fedora, possui interface gráfica e suporte para animação da simulação. A presença da interface gráfica é um fator facilitador, pois descomplica o processo de montagem da simulação. Em contrapartida, esse recurso é um limitador para testes de DDoS, pois reduz drasticamente o desempenho em presença de muitos nós.

O NCTUns possui alguns artigos científicos relacionados, porém a comunidade de desenvolvedores se mostra menor que a do NS-3. A documentação e a página oficial desse simulador também possuem qualidade inferior comparadas ao NS-3, dificultando o aprendizado e o uso dessa solução.

3.1.5. GTNetS (*Georgia Tech Network Simulator*)

O GTNetS é um ambiente de simulação para o estudo do comportamento de redes de escala média a grande. Seu projeto busca imitar o conceito real das redes, isto é, separar as camadas em componentes distintos.

O programa foi desenvolvido em C++ e possui extensa documentação sobre seu projeto. Seus principais recursos são: conexões orientadas a aplicações; suporte a animação; suporte a simulação distribuída; e geração de estatísticas de simulação.

O GTNetS implementa menos modelos de protocolos do que o NS-3, principalmente na camada de redes e na camada de enlace. Além disso, não há suporte para a importação de topologias.

3.1.6. IMUNES (*Integrated Multiprotocol Network Emulator / Simulator*)

O IMUNES, assim como o NCTUns, utiliza o *kernel* do sistema operacional para realizar suas simulações, no entanto, o IMUNES é baseado no FreeBSD. Essa característica lhe permite funcionar como emulador de redes também.

Além da interface gráfica, é possível utilizar uma interface de linha de comando, porém seu manual não explicita a possibilidade de interação com o código do usuário e nem de integração com outras ferramentas (KUKEC, 2011). Essa característica configura uma grande desvantagem, pois torna onerosa a integração desse simulador com outros programas.

3.2. NS-3 (*Network Simulator 3*)

O NS-3 é a terceira versão de um simulador de redes que vem sendo desenvolvido de forma colaborativa na Internet. Algumas empresas e instituições de ensino contribuem de maneira ativa para o seu desenvolvimento, podendo-se citar National Science Foundation, Planète Group, INRIA Sophia Antipolis, Georgia Institute of Technology, University of Washington, Universidade do Porto e Google.

O NS-3 é uma evolução do simulador NS2. O último utiliza a linguagem OTcl (*Object Tool Command Language*), sendo necessário empregá-la para executar uma simulação. O NS-3 foi escrito inteiramente em C++ e possui um suporte opcional a Python. Além disso, possui melhorias nos seguintes aspectos: simulação correta de nós com múltiplas interfaces; uso do endereçamento IP; maior consistência com arquiteturas e protocolos da Internet; maior detalhamento do modelo 802.11; e outras. (NS-3 PROJECT, 2012).

A cada três meses, a equipe de desenvolvimento do NS-3 lança uma nova versão do software, sendo a 3.13 a mais recente na data de produção deste trabalho. O NS-3 foi escolhido como plataforma para esse trabalho pelos seguintes motivos:

- Desempenho reconhecido pela comunidade científica;
- Presença de trabalhos anteriores e atuais no IME usando a plataforma;
- Documentação adequada;
- Comunidade grande e ativa de usuários; e
- Projeto desenvolvido com a licença GNU GPL, ou seja, código livre.

A seguir, será apresentada uma breve descrição do funcionamento desse simulador .

3.2.1. Arquitetura do NS-3

O NS-3 é um simulador de eventos discretos escrito em C++. Ele foi projetado como uma série de bibliotecas que podem ser referenciadas dinamicamente ou estaticamente em um programa principal. Este programa é encarregado de gerar a topologia e iniciar a simulação propriamente dita. O NS-3 também exporta quase todas as suas API para Python, permitindo que o programa principal também seja escrito nessa linguagem.

A Figura 3.1 a seguir mostra a divisão de componentes do NS-3.

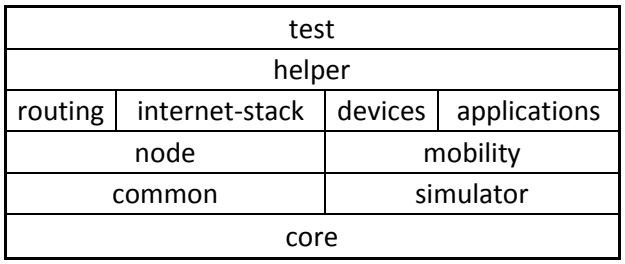

Figura 3.1 - Componentes do NS-3

Os componentes do NS-3 tem as seguintes funcionalidades:

- core – Esse componente está presente em todos os protocolos, hardware (simulado) e modelos de ambiente. O core implementa a maioria dos padrões de projeto do software e alguns recursos de *logging*.
- common – Suplementa a função do core no que diz respeito à função de *logging*. É responsável pelos pacotes e pela escrita em arquivos PCAP e ASCII.

- simulator – Gerencia e controla os eventos da simulação e realiza a aritmética de tempo.
- node – Representa os nós e os dispositivos de rede. Também implementa as classes de endereçamento.
- mobility - Implementa a movimentação física dos nós ao longo da simulação.
- routing – Implementa os protocolos de roteamento.
- internet-stack – Implementa a pilha de protocolos da internet.
- devices – Implementa os enlaces da rede.
- applications – Implementa algumas aplicações de uso comum na internet.
- helper – Fornece uma interface para o usuário facilitando a configuração de vários parâmetros de simulação através de sua interface.
- test – Conjunto de classes que verificam o funcionamento do simulador e o desempenho do computador.

Através dessa organização de componentes o NS-3 pretende fornecer ao usuário a capacidade de:

- Construir uma rede virtual com nós, enlaces, protocolos e aplicações da Internet ou de outro tipo de rede;
- Fornecer mecanismos para o controle de uma simulação de eventos discretos;
- Possibilitar a interação do simulador com a rede real;
- Suporte para animações de simulações de rede; e
- Possibilitar a geração de *traces*, arquivos de *log* e estatísticas a respeito dos resultados da simulação.

O NS-3 possui diversos padrões de projeto planejados para facilitar seu uso. Visando a reduzir o acoplamento entre as classes dos diversos componentes, utiliza-se a estrutura de *callback*. Para gerenciar a memória, é usada a contagem de referência dos objetos orgânicos do NS-3. A geração de variáveis aleatórias, essenciais em um simulador, é feita através do gerador MRG32k3a, capaz de gerar números aleatórios com períodos de aproximadamente $3{,}1 \times 10^{57}$.

4. Geradores de Topologia

A topologia de uma rede está diretamente relacionada ao seu desempenho. Informações como largura de banda, retardo e número de conexões podem ser extraídos da topologia. Essas informações têm influência direta sobre parâmetros de uma rede como *troughtput*, robustez e latência. Desta forma, a representação de uma topologia fiel à realidade trará resultados de simulação mais fidedignos.

O estudo de topologias na Internet é dividido em dois níveis: de sistemas autônomos (SAs) e de roteadores. O primeiro é obtido pelas listas de endereços dos roteadores de borda, que implementam o protocolo BGP (*Border Gateway Protocol*). O segundo nível, o de roteadores, utiliza as tabelas de endereço IP e representam as redes internas aos SAs. A separação dos níveis é explicada pelas diferenças nas regras de roteamento e estabelecimento de enlaces, os SAs possuem motivações comerciais e políticas.

Vários pesquisadores se dedicam a propor modelos matemáticos que possam reproduzir as características da Internet. O primeiro desafio desses pesquisadores é definir as métricas responsáveis por prover a similaridade adequada entre topologias geradas e reais. Algumas dessas métricas são:

- Grau médio – é a média dos graus dos nós de um grafo;
- Distribuição dos graus – a distribuição estatística dos graus dos nós de um grafo;
- Distribuição conjunta dos graus – distribuição estatística da soma dos graus de dois nós conectados por uma aresta;
- Agrupamento – é uma medida de quão perto os vizinhos de um nó estão de formar um clique (subgrafo completo);
- *Rich-club connectivity* – mede a hierarquia de um grafo, ou seja, uma relação entre os nós menos conectados com os nós mais conectados;
- *Coreness* – é uma medida de conectividade de um nó em relação a outros nós em função de seu grau;
- Distribuição de distâncias – é a distribuição de probabilidades das distâncias entre nós, naturalmente varia de um até o diâmetro do grafo;

- *Betweenness* – é uma medida de centralidade de um nó no grafo. Essa medida é obtida calculando o número de caminhos mínimos que passam pelo nó sobre o número de arestas que não contém esse nó; e
- Espectro – Conjunto de autovalores da matriz de adjacência.

As dificuldades de modelar a topologia da Internet começam em definir quais aspectos a caracterizam de maneira adequada e, por consequência, quais métricas empregar. Ainda, a modelagem isolada de uma dessas características não trará resultados eficientes, pois não há correlação explícita entre elas. A validação dessas propostas ainda está sujeita à correta coleta de dados da rede real. Há indícios de falhas no processo de aquisição dos mapas da Internet (CHEN, 2002).

A validação dos modelos criados se esbarra na ausência de dados que forneçam uma visão completa e precisa da Internet. Dados no nível de SA da topologia da Internet se encontram normalmente desatualizados. Já no nível de roteadores, os dados normalmente representam informações sensíveis para um ISP (*Internet Service Provider*) e, portanto, não são divulgados publicamente.

A tarefa de gerar grafos pseudo-randômicos não é trivial, existem diversos modelos que pretendem solucionar esse problema. A seguir, será apresentado um breve histórico dos modelos propostos para geração de topologias. Em seguida, serão descritos os geradores de topologias estudados, uma seção abordará o BRITE (*Boston University Representative Internet Topology Generator*) - ferramenta escolhida - e sua integração ao projeto.

4.1. Histórico dos Modelos de Geração de Topologias

As redes complexas como a Internet geralmente são descritas usando a teoria de grafos randômicos de Erdõs-Reny. Waxman apresentou uma variação dessa proposta que passou a ser o primeiro modelo para geração de topologias amplamente usado. A mudança consistia em alterar a probabilidade de adição de uma nova aresta ao grafo considerando a distância euclidiana.

Uma linha de pesquisa posterior observou que existia certa hierarquia entre os nós de uma rede. Foram introduzidos os conceitos de nó de trânsito (*transit*, grau maior do que um) e nó folha (*stub*, grau igual a um). A partir dessas idéias, surgiram os geradores de topologia Transit-Stub e Tiers. Esses geradores foram chamados estruturais ou hierárquicos.

Faloutsos apresentou um artigo que mostrava que o grau dos nós era bem representado por leis de potência (FALOUTSOS, 1999). Esse conceito foi amplamente discutido na comunidade cientifica e é, até hoje, bastante utilizado na geração de topologias.

Barabási apresentou um modelo que mistura leis de potência e uma adição seletiva de nós e arestas (BARABÁSI, 2002). Os nós a serem incrementados tem maior probabilidade de se conectar a nós bem conectados. Essa característica complementa o modelo de leis de potência com a idéia de hierarquia de ligação. Outros modelos de geração como o GLP (*Generalized Linear Preference*) foram idealizados com algumas variações das aplicações dos conceitos de leis de potência e hierarquia de conexão.

Alguns modelos mais recentes se utilizam de subgrafos de topologias reais conhecidas para gerar novas topologias. Outros modelos procuram implementar heurísticas de adição de nós. Essas propostas ainda não possuem análise detalhada que comprovem sua eficiência.

4.2. Descrição dos Geradores de Topologia Avaliados

As ferramentas de geração de topologia evoluíram com o entendimento da arquitetura da Internet. A determinação de métricas, a melhoria de modelos matemáticos e as inferências feitas sobre a topologia da Internet possibilitaram a validação dessas ferramentas. Desta forma, serão abordados alguns dos geradores mais empregados pela comunidade científica na presente seção.

4.2.1. Waxman

O gerador Waxman é o primeiro gerador de grafos aleatórios, sendo baseado no modelo de rede aleatória. Nele, os vértices são dispostos em um plano bidimensional através de um processo de distribuição de Poisson. Suas arestas são adicionadas obedecendo uma função probabilística dependente da distância euclidiana entre os vértices que liga.

Como entrada, são fornecidos os números de vértices e as coordenadas do espaço geográfico onde estes serão dispostos. Existe um valor D que determina a distância máxima entre dois vértices. Além disto, existe um parâmetro α diretamente proporcional à taxa de arestas longas e outro parâmetro β proporcional à densidade de arestas do grafo.

Em pouco tempo, ele foi substituído por geradores hierárquicos. Sua natureza aleatória revelou uma incapacidade de representar métricas importantes, como as leis de potência na distribuição dos graus, presentes na topologia da Internet em nível de SAs.

4.2.2. GT-ITM (ou gerador *transit-stub*)

Este gerador (*Georgia Tech Internetwork Topology Model*) tem como principal característica reproduzir a estrutura hierárquica da Internet, sendo chamado de modelo *transit-stub*. Como parâmetros, o usuário deve fornecer a quantidade de: *transit* e *stub domains*, LANs por *stub domain*, e arestas entre *transit* e *stub domains*.

Para isto, gera-se inicialmente um grafo conexo aleatório através do método Waxman ou de uma de suas variantes. Cada nó, que representa um *transit domain*, é expandido para gerar outro grafo conexo aleatório que será a topologia *bakcbone*. Em seguida, geram-se grafos aleatórios que farão o papel dos *stub domains*, sendo conectados nos vértices dos *transit domains*. A geração é concluída com a adição de algumas arestas extras entre vértices de um *transit domain* e outro de um *stub domain.*

Um gerador similar e com a mesma abordagem é o Tiers, entretanto seu modelo hierárquico se baseia em três níveis: WAN, MAN e LAN. Os seus subgrafos conexos

são gerados pela união dos vértices, em um único domínio, empregando um árvore geradora mínima.

4.2.3. Inet (*Internet Topology Generator*)

O Inet é um gerador de topologia à nível de SA. A topologia gerada pelo Inet está em conformidade com as tabelas BGP obtidas de *RouteViews* e *NLANR*, coletadas de 51 topologias entre novembro de 1997 e fevereiro de 2002. O número de nós mínimo é de 3037 (quantidade de SAs existentes na época) e a fração de nós com grau um respeita o valor de 0,3 extraído das tabelas BGP (WINICK, 2002). Estes dados, entretanto, forneciam apenas a conectividade entre os vértices por meio de uma matriz de adjacência. Desta forma, em sua versão 2.2, o presente gerador não considerou características importantes como largura de banda e latência.

Para a geração do grafo, segue-se uma lei de potência com corte exponencial (redutor do comportamento de cauda pesada[2]) (CHAKRABARTI, 2006), atribuindo um grau a cada vértice. Para os vértices com grau maior que um, gera-se uma árvore geradora mínima. Esta restrição serve para garantir apenas um componente conexo. Agora, os vértices de grau um são conectados a estas árvores geradas por meio do método da conexão preferencial. Da mesma forma, começando pelos vértices de grau mais elevado, inserem-se as arestas das árvores geradoras mínimas até cada vértice atingir o seu grau inicialmente especificado.

Em sua versão 3.0, para evitar conectar vértices de grau baixo, a função que gera as arestas recebeu um peso dependente dos graus dos vértices ligados. Esse peso varia quase linearmente para vértices com graus parecidos, favorecendo a ligação de vértices com maior diferença de graus.

De toda forma, algumas métricas de conectividade (tamanho máximo da clique e o coeficiente de agrupamento) não são satisfeitas (WINICK, 2002), sugerindo a necessidade de melhor entender a conectividade da Internet.

[2] Como uma distribuição de cauda pesada não possui a área sob a curva limitada por uma função exponencial, a adição do corte exponencial ameniza a ausência dessa função de limite, diminuindo o comportamento de cauda pesada.

4.3. BRITE (*Boston University Representative Internet Topology Generator*)

A tarefa de desenvolver um gerador de topologia que atenda todos os requisitos de representatividade, flexibilidade, extensibilidade e eficiência é extremamente árdua. Manter grafos gigantescos, respeitando características topológicas inferidas da análise de dados coletados se esbarra em limitações computacionais e matemáticas. Desta forma, um gerador de topologia não pode atender todos os requisitos possíveis.

O BRITE, nesse cenário, se destaca pela sua universalidade, fazendo-o ser empregado e citado em importantes pesquisas na área de redes complexas. Mesmo com as suspeitas de que a observação de leis de potência são conseqüência da falta de dados que melhor representem a topologia da Internet (CHEN, 2002), são poucas as novas propostas para substituir essas leis. Em contrapartida, estudos defendem os resultados obtidos pelas leis de potência (TANGMUNARUNKIT, 2002).

As ferramentas existentes que alegam propor melhores soluções para os geradores de topologia não estão disponíveis para a comunidade científica. As solicitações feitas para obter acesso ao código dessas ferramentas, como a desenvolvida pelo bem conceituado CAIDA, foram negadas.

Outro ponto interessante em fornecer suporte ao BRITE é que, transitivamente, incorporam-se diversos outros formatos de topologia ao NS-3. Assim, o presente trabalho agrega, indiretamente, maior abrangência ao simulador empregado.

Vale ainda citar que, entre os geradores estudados, o BRITE é o único que possibilita a geração de topologias em dois níveis. Este tipo de topologia melhor aproxima características relacionadas ao grau e à hierarquia de uma rede, representando mais adequadamente a Internet.

Por fim, a documentação do gerador BRITE disponível é vasta. Outro aspecto a ser considerado é sua abertura à ampla customização dos parâmetros de configuração. Estudos voltados ao aprimoramento destes dados de entrada exibem resultados satisfatórios para as topologias geradas (UHLIG, 2008).

4.3.1. Principais Características

O gerador BRITE produz grafos em grande escala utilizando leis de potência, podendo ser empregado para gerar topologias em nível de SA e de roteador. Além disto, ele é capaz gerar topologias de duas camadas, ou seja, abrangendo esses dois níveis simultaneamente. Uma de suas inovações é a agregação de dados importantes para o estudo da rede, por exemplo: a largura de banda, os identificadores de SA e o tempo de atraso.

Como esse gerador foi desenvolvido para ser universal, ele não está fortemente acoplado a quaisquer modelos específicos. Desta forma, o BRITE pode ser facilmente adaptado para diversos cenários mantendo a representatividade de seus resultados. Ainda existe uma boa recepção ao desenvolvimento de novos modelos.

Além da larga flexibilidade permitida pelo grande número de parâmetros de configuração, o BRITE possui compatibilidade com outros geradores de topologia (GT-ITM e Inet) e visualizadores de grafo. Existe ainda um GUI (*Graphical User Interface*) para facilitar a criação dos arquivos de configuração para as topologias.

O BRITE ainda permite importar topologias obtidas pela coleta de informações em redes reais, como as dos projetos NLANR e Skitter. Estas topologias são então construídas e podem ser combinadas com outras geradas pelo BRITE.

Para fins de modularização, o BRITE se divide em duas grandes *engines*: a de geração e a de análise. A primeira é responsável por ler os parâmetros de configuração ou os arquivos importados e transformá-los em um grafo que pode ser exportado em diversos formatos. A última, chamada de BRIANA (*BRITE Analysis Engine*), fornece um conjunto de operações de análise. Entre elas, a freqüência de grau de saída e a distribuição de tamanho de caminho, que auxiliam o processo de criação da topologia. A Figura 4.1 apresenta o esquemático estrutural do BRITE.

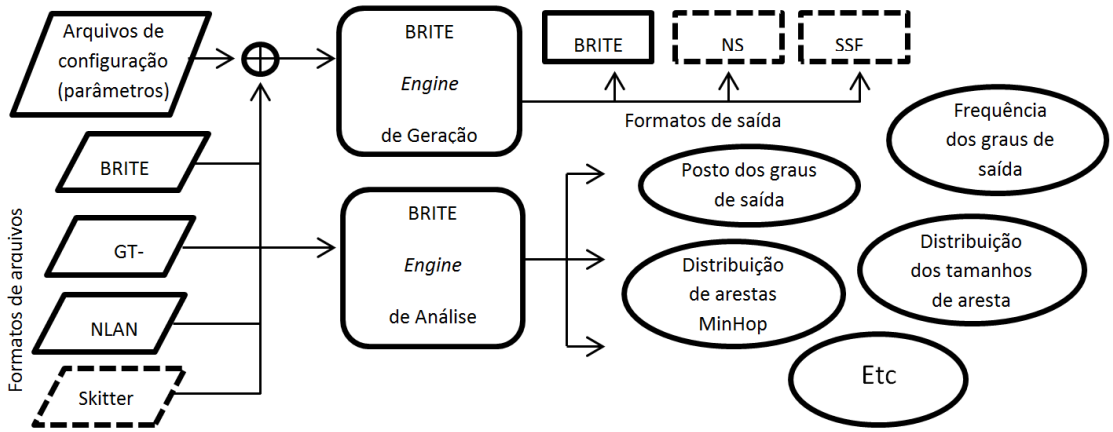

Figura 4.1 – Esquemático estrutural do BRITE (MEDINA, 2001)

A distribuição de vértices no BRITE é feita em um plano cartesiano, podendo ser realizada aleatoriamente ou de acordo com uma função de cauda pesada (gerando agrupamentos). De acordo com o modelo empregado, o número de arestas por vértice pode ser um parâmetro informado pelo usuário. O crescimento da rede se dá de acordo com dois modelos: Waxman e Barabási-Albert. Já a interconexão dos vértices pode ser das seguintes formas: escolha aleatória, conexão preferencial ou conexão preferencial ponderada.

Os detalhes de como se dá o processo de geração da topologia estão intimamente relacionados ao modelo empregado, ou seja, não há uma divisão exata das etapas envolvidas no processo. Pode-se, entretanto, dizer que os vértices são dispostos no plano cartesiano e, em seguida, são interligados pelas arestas. Após isso, são atribuídos dados sobre a topologia aos componentes do grafo. Por fim, a topologia gerada é transformada em um formato específico de exportação.

4.3.2. Modelos Implementados

Em sua distribuição, o BRITE possui integrado oito modelos de geração distintos. Além dos modelos Waxman e Barabási-Albert (BA) para os níveis de SA e de roteador, pode-se importar arquivos dos formatos BRITE, GT-ITM, NLANR, Inet e

Skitter. Pode-se também gerar topologias de dois níveis (SAs e roteadores) com os modelos hierárquicos *top-down* e *bottom-up*.

Apesar do BRITE já fornecer estes modelos, não existe qualquer imposição sobre o uso dos mesmos. O usuário pode combinar os diversos modelos existentes e, até mesmo, criar novos a partir do zero, caso eles não atendam os requisitos desejados.

4.3.2.1. Modelos a nível de roteador

Nesta divisão, enquadram-se os modelos Waxman e Barabási-Albert para nível de roteador. O BRITE separa a disposição dos vértices dos processos de crescimento da topologia e de conexão dos vértices. Estes dois processos são a principal diferença entre os dois modelos citados.

Os nós podem ser dispostos: aleatoriamente ou de acordo com uma distribuição de cauda pesada. Segundo Medina, importantes características da Internet seguem esta última distribuição (MEDINA, 2001). Ao optar pela disposição seguindo cauda pesada, o plano cartesiano é dividido em quadrados. Cada um destes recebe um número de nós (esta quantidade segue uma distribuição de mesma natureza), que são distribuídos aleatoriamente nessa sub-região. Vale ressaltar que os nós dispostos não necessariamente farão parte da topologia, pois podem ser descartados no processo de crescimento da rede.

Com a topologia montada, são atribuídos os valores de largura de banda para as arestas. Essa atribuição respeita três parâmetros de configuração: *BWdist*, *BWmin* e *BWMax*. O primeiro define qual dos seguintes tipos de distribuição será seguido: constante, uniforme, exponencial ou cauda pesada.

O modelo Waxman de conexão dos vértices já foi explicado na Seção 4.2.1. Já no modelo BA, a implementação que acompanha o BRITE opta pela técnica do crescimento incremental.

4.3.2.2. Modelos a nível de SA

Esta divisão engloba os modelos Waxman e Barabási-Albert para nível de SA. Eles são muito parecidos com os modelos correspondentes descritos na Seção 4.3.2.1. A diferença agora é que os vértices são tratados como sistemas autônomos, possibilitando que eles contenham topologias associadas.

4.3.2.3. Modelos Hierárquicos

A fidelidade a aspectos estruturais da Internet são significativos para atingir um modelo que reflita da melhor maneira possível a realidade. De toda forma, atingir essas características não pode prejudicar a boa representatividade já alcançada seguindo propriedades relacionadas ao grau.

Haddadi (2008) afirma que, até o momento de sua publicação, inexiste uma ferramenta que consiga integrar essas características hierárquicas e relacionadas ao grau de maneira adequada. Ferramentas que parecem caminhar nesse sentido, como o gerador de topologia CAIDA (*The Cooperative Association for Internet Data Analysis*), não estão disponíveis para o público.

O BRITE possibilita gerar topologias hierárquicas de dois níveis, seguindo as seguintes abordagens: *top-down* e *bottom-up*. Se empregada recursão, pode-se alcançar 2×n camadas em n passos.

Na abordagem *top-down*, o BRITE gera primeiramente uma topologia em nível de SA empregando algum de seus modelos. Em seguida, para cada SA, gera-se uma topologia em nível de roteador. A ligação das topologias em nível de roteador obedecendo a conectividade da topologia de SAs é um área que ainda precisa ser estudada mais a fundo (HADDADI, 2008). Como alternativa, o BRITE emprega os quatro modelos herdados do conhecido gerador GT-ITM.

A largura de banda associada é controlada por parâmetros que regem as distribuições para arestas inter e intra-domínios, ou seja, eles sobrepõe os parâmetros de cada nível hierárquico.

A abordagem *bottom-up* adotada pelo modelo não foi validada e, portanto, não será abordada de maneira detalhada no presente estudo. A título de conhecimento, gera-se uma topologia em nível de roteador. Em seguida, o modelo designa um número de vértices para cada SA e, por fim, agrupa-os em seus respectivos SAs.

4.3.2.4. Modelos Importados

É possível importar topologias do próprio BRITE, do GT-ITM, do NLANR, do Inet e do CAIDA Skitter. Estes formatos são convertidos para uma topologia nativa, como se tivesse sido gerada pelo próprio BRITE. Com este recurso, faz-se possível combinar diversos modelos e topologias já existentes de outros projetos.

5. Geração de Tráfego de Fundo

Um ataque distribuído de negação de serviço tem como um de seus objetivos principais negar um recurso a um usuário legítimo do sistema alvo. Para atingi-lo, o comportamento desses usuários deve ser simulado em toda a rede. Trata-se do tráfego de fundo, que é composto por esse fluxo de dados legítimo.

A análise do tráfego incidente no nó alvo fornece dados para avaliar a efetividade de um ataque. Com essas informações, é possível determinar o número de usuários legítimos atendidos durante um ataque e a perda de pacotes, obtendo, assim, dados sobre a eficiência do ataque DDoS.

5.1. Requisitos para um modelo de geração de tráfego de fundo

Diversos modelos são utilizados para gerar sequências de pacotes que sejam próximas ao fluxo real de pacotes da Internet. Esses modelos devem possuir as propriedades estatísticas das redes reais e devem ser simples o bastante para minimizar os custos de memória e tempo computacional (AMMAR, 2011).

Um processo estocástico destinado a simulação do tráfego de fundo deve obedecer aos seguintes requisitos (ZUKERMAN, 2003):

i. Ser definido por um pequeno número de parâmetros.
ii. Caso esses parâmetros sejam definidos utilizando estatísticas reais, devem ser observados:
    a. O tráfego simulado deve possuir as mesmas características estatísticas do tráfego real, incluindo a função de autocovariança do processo estocástico.
    b. Caso a fonte represente uma fila de um único servidor (*Single Server Queue - SSQ*), o desempenho do modelo deve prever com precisão o tráfego real em SSQ reais.
iii. Deve ser de fácil análise.

Originalmente, o tráfego da Internet foi gerado utilizando processos estocásticos sem memória. Estudos recentes indicam três características fundamentais que representam um fluxo de dados da rede, são elas: cauda pesada, *Long-Range Dependence (LRD)* e *Self-Similarity* (ZUKERMAN, 2003).

Distribuições de cauda pesada possuem grande variação na amplitude de dados com probabilidades altas em valores distantes da média (RESNICK, 2007). A característica de LRD mostra que o processo não é sem memória, pois existem padrões que se repetem ao longo de grandes intervalos. Já a de *Self-Similarity* indica que o processo possui o mesmo comportamento independentemente da escala e do tempo em que está sendo utilizado (KARAGIANNIS, 2004).

A simulação de um ambiente de Processo de Rajadas de Poisson Pareto (*Poisson Pareto Burst Process* - PPBP) reproduz essas características, atendendo aos requisitos estabelecidos. Esse modelo vem sendo utilizado para geração de tráfego de fundo em simulações de rede (AMMAR, 2011).

5.2. Processo de Rajadas de Poisson Pareto

O PPBP (*Poisson Pareto Burst Process*) foi proposto para ser um modelo mais realístico do tráfego da Internet do que seus predecessores. Os eventos nesse processo representam pontos no tempo em que diversas fontes iniciam ou terminam de transmitir dados.

De acordo com esse modelo, rajadas de dados são geradas com uma distribuição de Poisson de parâmetro λ. O tamanho das rajadas segue uma distribuição de Pareto de parâmetro H, e cada uma delas é transmitida segundo uma taxa fixa. A qualquer momento, é possível que uma nova fonte inicie a transmissão enquanto outras estão ativas. O PPBP também é chamado M/Pareto/∞ ou simplesmente M/Pareto (ZUKERMAN, 2012).

O presente trabalho irá utilizar uma implementação do PPBP fornecida por Doreid Ammar e Thomas Begin. O módulo foi desenvolvido para NS-3 e possui um artigo explicativo com a respectiva validação dos resultados (AMMAR, 2011).

A Tabela 5.1 descreve os parâmetros de configuração da ferramenta.

Parâmetro	Descrição
$\boldsymbol{H}$	Parâmetro de Hurst definido para a distribuição de Pareto.
$\boldsymbol{T_{on}}$	Média da distribuição de Pareto.
$\boldsymbol{r}$	*Bit-rate* da rajada.
$\boldsymbol{E[n]}$	Número médio de rajadas ativas
$\boldsymbol{Tamanho}$	Tamanho dos pacotes

Tabela 5.1 – Parâmetros da ferramenta PPBP.

6. Ferramenta de Simulação

A montagem de um ambiente de simulação para um ataque distribuído de negação de serviço emprega todos os conceitos apresentados nos capítulos anteriores. Para automatizar a construção desse ambiente, foi desenvolvida uma ferramenta. Nesse capítulo, serão descritas as etapas realizadas por esta ferramenta para a criação de uma instância desse ambiente.

Primeiramente, abordar-se-á a conversão dos elementos da topologia do gerador BRITE para o NS-3. Em seguida, a topologia propriamente dita será montada, incluindo a distribuição dos endereços IP. Com o estabelecimento do roteamento entre os nós da rede criada, são instaladas as aplicações para a geração do tráfego de fundo. Nesta etapa, são definidas as portas das aplicações e o tempo que cada nó folha permanecerá ligado. Em paralelo, alguns dos nós folhas são escolhidos para se tornarem membros da *botnet*. Por fim, instalam-se as aplicações para o ataque distribuído de negação de serviço nesses nós escolhidos.

O presente capítulo se limita a descrever qual a metodologia empregada pela ferramenta para construir uma instância de simulação. Os procedimentos necessários para realizar a simulação propriamente dita, ou seja, o uso da ferramenta, serão descritos no Anexo C.

6.1. Conversão de topologia BRITE para NS-3

Devido ao grande número de referências ao gerador BRITE em pesquisas sobre topologia, esforços no sentido de integrá-lo a um dos mais famosos simuladores de rede, o NS-3, foram realizados por diversos grupos. Das contribuições mais significativas encontradas, destaca-se uma *branch* de uma versão de desenvolvimento do NS-3, não incorporada à *release* seguinte do simulador.

Em seu curso, o presente trabalho observou instabilidades da versão do NS-3 sobre a qual a *branch* havia sido criada, além da conseguinte perda dos recursos incorporados a versões mais recentes do simulador. Dentro das falhas encontradas,

destaca-se a impossibilidade do pyViz, um visualizador de topologias, gerar a simulação com auxílio gráfico a partir da topologia fornecida.

Desta forma, concentraram-se esforços em incorporar a integração entre o BRITE e o NS-3 para a última versão (3.13) disponível do simulador. Com esta integração, realizou-se a conversão da topologia do formato gerado pelo BRITE para o formato do NS-3. O trabalho desenvolvido será descrito a seguir.

Inicialmente, baixou-se a última versão disponível do BRITE, compilando-o para torná-lo disponível para a integração. Em seguida, a *branch* encontrada com a integração foi copiada para o diretório de módulos do NS-3. Neste módulo (chamado "brite"), existe uma classe *helper* responsável pela transformação de uma topologia em formato BRITE para duas listas de *structs*: uma contém informações sobre os nós e a outra sobre os vértices.

Informações geradas pelo BRITE	
Vértices	**Arestas**
Identificador	Identificador
Coordenada X	Vértice de origem
Coordenada Y	Vértice de destino
Grau de Entrada	Distância
Grau de Saída	Atraso
Identificador de SA	Largura de banda
Tipo de vértice	SA de origem
	SA de destino
	Tipo de aresta

Figura 6.1 – Composição das *structs* geradas à partir da topologia BRITE

O processo de transformação se dá por meio de três passos. No primeiro, obtém-se a topologia BRITE informando a localização de três arquivos: configuração, semente a ser empregada e nova semente gerada. Originalmente, o BRITE sobrepõe a semente atual com uma nova a cada vez que é executado. Este comportamento foi removido, visto que impossibilita que a simulação seja executada identicamente diversas vezes. No

segundo passo, todas as informações sobre cada vértice são transformadas em *structs* e salvas em uma lista. O mesmo é feito para todas as arestas no último passo. A composição de cada *struct* é mostrada na Figura 6.1.

Como o NS-3 emprega o Waf para configuração, compilação e instalação de suas simulações, é necessário criar um *script* que execute estas tarefas para o módulo de integração adaptado. Nesta etapa, foi preciso atentar para as mudanças de formato e de comandos entre as versões.

Para a conversão propriamente dita, faz-se necessário a correta utilização da lista de vértices e arestas geradas pelo *helper* descrito anteriormente. Para isto, deve-se criar um *NodeContainer* global para inserção dos vértices. Posteriormente, ligam-se os vértices, atribuindo as características de cada conexão. Isto é feito empregando um *PointToPointHelper*, adicionando as instalações feitas a um *NetDeviceContainer* global, responsável por todas as interfaces da rede. Nesta etapa, criam-se diversas instâncias de N*odeContainer*: uma para cada SA, uma para o conjunto de nós folhas e uma para os nós que interligam os SAs distintos. Do mesmo modo, são criadas diversas instâncias de *NetDeviceContainer*: uma para cada conjunto de interfaces internas de um SA e uma para as interfaces que ligam SAs distintos.

Para que o visualizador pyViz distribua fisicamente (em termos de coordenadas cartesianas) os vértices de maneira idêntica ao que foi gerado pelo BRITE, deve-se empregar um *MobilityHelper*. Com este *helper*, opta-se pelo modelo de posição constante, extraindo as coordenadas da lista de vértices citada anteriormente.

6.2. Atribuição de Endereços IP

O BRITE fornece a classificação dos nós, possibilitando uma melhor segmentação das redes que compõe cada SA. Apesar disto, o NS-3, na versão utilizada pelo presente trabalho, não apresenta implementação de protocolos de roteamento *classless*, como o OSPF (*Open Shortest Path First*) e o BGP (*Border Gateway Protocol*). Deste modo, a distribuição de endereços IP e de máscaras conforme a exigência desses protocolos torna-se desnecessária.

Desta forma, optou-se pela simplificação da distribuição de endereços. Além disto, como o IPv4 ainda representa a maioridade do tráfego existente (MAWI, 2012) e o NS-3 não possui algumas funcionalidades desenvolvidas para IPv6 (NS-3 PROJECT, 2012), será descartada esta última versão do protocolo IP.

Para cada conjunto de interfaces internas de um SA pertencente à topologia, será reservada uma faixa contínua de endereços, partindo do endereço base 200.0.0.0. Para as interfaces que ligam SAs distintos, a faixa contínua terá 10.0.0.0 como endereço base. A divisão desses grupos de interfaces já foi feita por meio da criação de instâncias de *NetDeviceContainer*, conforme Seção 6.1.

Em ambas situações, o procedimento adotado é o mesmo. Contam-se quantas interfaces existem em uma instância de *NetDeviceContainer*. Com esse valor, pode-se calcular quantos *bits* são necessários para se reservar uma faixa contínua de endereços para o grupo, este valor é denominado n. A máscara de rede é calculada a partir de n. Para obter o endereço inicial de cada grupo, soma-se 2^{n-1} ao seu endereço base. Com esses dados, a atribuição é feita por meio de um *Ipv4AddressHelper*.

6.3. Roteamento

Como dito na Seção 6.2, a versão utilizada do NS-3 não suporta os protocolos de roteamento OSPF, que seria empregado dentro de cada SA, e o BGP, que seria empregado para a comunicação entre SAs distintos. Como não existem alternativas equivalentes implementadas, optou-se por utilizar a única alternativa de roteamento viável oferecida pelo NS-3.

O protocolo de roteamento escolhido utiliza menor distância entre dois vértices, aproveitando-se do fato de conhecer toda a topologia. O algoritmo utilizado para o cálculo dessa distância é o de busca em largura (BFS – *Breadth First Search*), cuja complexidade é um fator impeditivo para redes de magnitude das que se desejam simular (NS-3 PROJECT, 2012). A montagem das tabelas de roteamento geraria uma demanda muito elevada de esforço computacional e de espaço de armazenamento

Para contornar essa limitação, o NS-3 oferece o *NIX Vector Routing*, cuja abordagem é idêntica, mas o cálculo das rotas é feito sob demanda. Este roteamento é instalado por meio da classe *helper Ipv4NixVectorHelper*.

6.4. Tráfego de Fundo

Antes de se abordar a geração do tráfego de fundo propriamente dito, precisa-se instalar em todos os nós folhas sorvedouros de pacotes TCP e UDP. Estes nós estão contidos em uma mesma instância de *NodeContainer*. Esta instalação se faz necessária para simular a recepção dos pacotes por alguma aplicação do nó. Como maioria substancial do tráfego atual é TCP/UDP (MAWI, 2012), outros tipos são desconsiderados.

Com a recepção do tráfego assegurada, precisa-se definir quais e quantas aplicações farão o envio e qual será o destino. A ferramenta recebe como parâmetro qual a probabilidade do tráfego gerado ser TCP ou UDP. Já as aplicações são determinados por suas portas, recebidas como parâmetro pela ferramenta por meio de dois vetores com portas TCP e UDP. Além disso, cada porta tem uma probabilidade associada, também recebida como parâmetro. A quantidade de aplicações por nó varia entre um valor mínimo e um máximo. Definido quantas e quais as aplicações a serem criadas, sorteia-se uma interface de destino para cada.

Para cada aplicação, cria-se uma instância da classe do *PPBPHelper*. Esta classe implementa o modelo de geração de tráfego de fundo escolhido pelo presente trabalho, conforme Seção 5. Os parâmetros desse gerador de tráfego de fundo também são determinados por meio de parâmetros passados à ferramenta.

Falta apenas determinar o tempo de vida de cada nó. Assim como o tempo total de duração da simulação, é passada à ferramenta uma lista de percentagens de duração (em relação ao tempo total) e as suas respectivas probabilidades de ocorrência. Com esta decisão feita, é sorteado um tempo de início para as aplicações do nó de modo que todas as aplicações rodem por completo, ou seja, terminem antes que a simulação atinja o seu tempo total. Utiliza-se a classe *ApplicationHelper* para configurar esse intervalo e para instalar as aplicações.

Todas essas decisões são feitas utilizando uma distribuição uniforme, tendo em vista que todos esses dados tem origem meramente estatística. De toda forma, a modificação do tipo de distribuição da variável aleatória é um processo relativamente simples.

6.5. Botnet

Para a *botnet*, necessita-se de um conjunto de máquinas infectadas e das aplicações específicas para o ataque instaladas nelas. Para que haja efetividade do ataque, ele deve ser coordenado de modo a ocorrer simultaneamente, ou seja, todas as aplicações devem iniciar, quase que ao mesmo instante, o envio de pacotes para o alvo.

A ferramenta conhece o alvo, a duração do ataque, o horário de início do ataque, o percentual de máquinas infectadas, as aplicações de ataque que serão executadas e os parâmetros destas. Esses dados são passados para a ferramenta pelo usuário.

Ao ser escolhido como "nó infectado", o nó recebe a instalação das aplicações de ataque escolhidas e tem seu tempo de início determinado. A aleatoriedade dessa escolha foi justificada na Seção 2.6. A ferramenta disponibiliza duas das quatro aplicações de ataque mais frequentes empregadas nas *botnets*: *UDP Flood* e *ICMP Flood* (PROLEXIC, 2012).

Como todos os nós já possuem um tempo de vida, determinado na Seção 6.4, e o ataque precisa ser simultâneo, o início do tempo de vida desses nós pode ser atrasado ou adiantado. Isto é feito de modo a obrigar que todos os nós infectados iniciem o ataque quase que simultaneamente.

7. Simulação Realizada

Na Seção 6, foi explicado o funcionamento da ferramenta desenvolvida para criar um ambiente de simulação de um ataque distribuído de negação de serviço. Em cada etapa abordada, pode-se observar a necessidade de um conjunto de parâmetros. À despeito das limitações já explicitadas ao longo do presente trabalho, a escolha desses parâmetros define a proximidade entre a simulação e a realidade. Desta forma, quão mais corretos forem, melhor será o resultado obtido.

A literatura estudada para o desenvolvimento desse trabalho especifica alguns dos valores desses parâmetros. Alguns outros, entretanto, por não terem sido encontrados estudos específicos a respeito de seus valores, foram escolhidos pelos autores do presente projeto a fim de possibilitar a execução de uma instância da simulação.

7.1. Modelo de Topologia

O ataque distribuído de negação de serviço pode envolver milhões de nós. Para recriar uma topologia que seja razoavelmente semelhante ao mapa da Internet, é necessário escolher o modelo mais adequado de geração de topologia. Essa escolha não é trivial, pois, conforme descrito nas seções anteriores, as métricas de comparação e avaliação de topologias constituem um assunto complexo ainda em estudo pela comunidade científica.

Haddadi (2008) realizou uma comparação entre os modelos BA, GLP, Waxman e Inet. Sua pesquisa também verificou os parâmetros ótimos para os algoritmos correspondentes aos modelos. Os resultados indicam que o BA possui melhor desempenho para geração de topologias em nível SA.

Tangmunarunkit (2002) compara os modelos randômicos (Waxman), hierárquicos (Tiers e Stub) e os baseados em grau dos nós (no caso, BA). Esse autor conclui que o modelo BA é o mais adequado para geração de topologias em nível SA e nível de roteadores.

Não existe um consenso quanto ao modelo mais representativo da Internet, pois as métricas de avaliação ainda não estão consolidadas. É possível encontrar artigos, como o de Haddadi (2008), que atribuem bons resultados para o modelo BA, enquanto outros, como Hernandez (2006), que não o fazem.

Autores como Lu (2011), Maciel (2008) e Magoni (2001) propõem novos modelos de geração de topologias que ainda não possuem implementação em domínio público. Isso dificulta seu emprego no presente projeto, dada a ausência de integração ao gerador de topologias e ao simulador. Além disso, faltam maiores estudos sobre essas novas propostas.

O modelo BA é orgânico ao BRITE e possui resultados reconhecidos para o nível de SA por diversos artigos, sendo, portanto, o adotado nesse trabalho. Como já foi apresentado, a geração de topologias para o nível de roteadores se esbarra na ausência de dados completos para a validação correta de modelos propostos. Haddadi (2008) afirma que, nesse nível, as leis de potência não são adequadas. Dessa forma, será utilizado um modelo aleatório: o Waxman. A construção da topologia se dará com a abordagem *top-down*, descrita na Seção 4.3.2.3. O tamanho do grafo precisa ser da ordem de dezena de milhares para o bom comportamento do gerador BRITE (MEDINA, 2001).

Os valores adequados dos parâmetros escolhidos para os modelos empregados foram determinados por Uhlig (2008). Vale lembrar que esses parâmetros em específico são do próprio BRITE, devendo ser inseridos, conforme o Manual do BRITE, em um arquivo de configuração à parte. Para a ferramenta, deve ser fornecido apenas o nome desse arquivo, além de outros dois arquivos de *seed* do BRITE, um com a semente atual e outro para armazenar a nova semente gerada.

7.2. Aplicações de Tráfego de Fundo

O MAWI (*Measurement and Analysis on the WIDE Internet*) *Working Group* é responsável pela medição e análise de tráfego em um *link* transoceânico entre os Estados Unidos e o Japão. Em sua página, esse grupo disponibiliza, desde julho de

2006, estatísticas diárias sobre esse tráfego. Pela sua representatividade, suas estatísticas foram empregadas para definir alguns parâmetros das aplicações de tráfego de fundo.

Analisando dados referentes ao mês de fevereiro de 2012, pode-se observar que, Figura 7.1 do tráfego UDP ou TCP existente, o primeiro protocolo é responsável por 18,8% do total. Desta forma, a fração de aplicações TCP existente na simulação foi definida como 81,2%. Na Figura 7.1, apresentam-se as estatísticas resumidas para o dia 10 de fevereiro de 2012.

protocol	packets		bytes		bytes/pkt
ip	49975350	(99.42%)	24281834949	(99.14%)	485.88
tcp	29880674	(59.45%)	21395868219	(87.35%)	716.04
http(s)	13050419	(25.96%)	17733906517	(72.40%)	1358.88
http(c)	10088271	(20.07%)	1222292507	(4.99%)	121.16
squid	59517	(0.12%)	32211978	(0.13%)	541.22
smtp	111423	(0.22%)	47116347	(0.19%)	422.86
ftp	26974	(0.05%)	2769888	(0.01%)	102.69
pop3	4712	(0.01%)	2102770	(0.01%)	446.26
imap	3100	(0.01%)	921066	(0.00%)	297.12
telnet	79563	(0.16%)	5905015	(0.02%)	74.22
ssh	375145	(0.75%)	59574992	(0.24%)	158.81
other	5996980	(11.93%)	2261600406	(9.23%)	377.12
udp	3624860	(7.21%)	1526512236	(6.23%)	421.12
dns	586657	(1.17%)	139841615	(0.57%)	238.37
realaud	9	(0.00%)	1147	(0.00%)	127.44
halflif	57	(0.00%)	4584	(0.00%)	80.42
starcra	54	(0.00%)	5081	(0.00%)	94.09
everque	3652	(0.01%)	2823475	(0.01%)	773.13
unreal	22	(0.00%)	3287	(0.00%)	149.41
quake	33	(0.00%)	3230	(0.00%)	97.88
cuseeme	18	(0.00%)	1330	(0.00%)	73.89
other	3034312	(6.04%)	1383800945	(5.65%)	456.05
icmp	15606418	(31.05%)	994660916	(4.06%)	63.73
ipip	185	(0.00%)	19270	(0.00%)	104.16
ipsec	26470	(0.05%)	9013652	(0.04%)	340.52
ip6	831051	(1.65%)	354705168	(1.45%)	426.82
other	5692	(0.01%)	1055488	(0.00%)	185.43
frag	450	(0.00%)	428302	(0.00%)	951.78

Figura 7.1 – Tráfego do dia 10 de fevereiro de 2012 no *link* da MAWI.

Para os mesmos dados, observou-se qual a contribuição em termos percentuais de cada aplicação. A Tabela 7.1 traz uma relação dessas observações. Para resumir as informações de tráfego, algumas aplicações foram agrupadas sob o nome de “outros”, sendo designado um valor fictício de porta para elas. Essas informações foram passadas como parâmetro para a ferramenta.

TCP		UDP	
Aplicação (Porta)	**Contribuição**	**Aplicação (Porta)**	**Contribuição**
HTTPS (443)	43,5%	DNS (53)	15%
HTTP (80)	33,8%	Half-Life (12488)	0,01%
Squid (1935)	0,24%	Starcraft (32064)	0,01%
SMTP (25)	0,44%	Unreal (36457)	0,01%
FTP (21)	0,10%	Quake (42552)	0,01%
POP3 (110)	0,02%	Cuseeme (32222)	0,01%
IMAP (143)	0,02%	Outros (2000)	84,95%
Telnet (23)	0,32%	-	-
SSH (22)	1,5%	-	-
Outros (1000)	20,06%	-	-

Tabela 7.1 - Tráfego das aplicações para o mês de fevereiro de 2012 no *link* da MAWI.

Ainda resta definir quanto tempo cada nó ficará ativo, ou seja, com suas aplicações enviando dados. Apenas 10% dos nós ficarão disponíveis durante toda a simulação. Do restante, 30% somente por 8,34% do tempo total de simulação e os 60% restantes, por 25% do tempo total (DAGON, 2007).

7.3. Formação da *botnet*

As aplicações de ataque a serem executadas pelos nós infectados serão as duas disponibilizadas pela ferramenta desenvolvida no presente trabalho, sem limite do número de pacotes UDP a serem enviados (SOUZA e JÚNIOR, 2011). Já a proporção de nós infectados será de 25% (ASSADHAN, 2009).

7.4. Parâmetros não encontrados na literatura

Como dito anteriormente, alguns parâmetros da simulação não estão explicitados na literatura, sendo necessário escolher um valor para eles. Além destes, existem parâmetros que não influenciam na fidelidade da simulação em si.

A Tabela 7.2 ilustra esses parâmetros e os respectivos valores escolhidos pelo autores do presente trabalho.

Parâmetro	Valor	**Parâmetro**	Valor
Nó alvo	Primeiro nó folha	Gerar PCAP	Sim
Porta de ataque	28	Gerar Visualização	Sim
Duração da simulação (em segundos)	100	Mínimo de aplicações por nó	1
Início do ataque (em segundos)	10	Máximo de aplicações por nó	5
Duração do ataque (em segundos)	50	Número médio de rajadas ativas do PPBP	10
Bit-rate da rajada PPBP (em Mbps)	1	Média da distribuição de Pareto	0.2
Tamanho do Pacote PPBP (em bytes)	100		

Tabela 7.2 – Parâmetros utilizados

8. Resultados

Com a simulação realizada no Capítulo 7, foram coletados os pacotes no nó alvo por meio de arquivos PCAP. Foram executadas simulações com três sementes distintas. Para cada semente, realizou-se a comparação entre o tráfego com e sem o ataque distribuído de negação de serviço.

É importante ressaltar que os dados apresentados não são necessariamente realistas, tendo em vista que a escolha de alguns parâmetros para a simulação não foram encontrados na literatura. Os resultados obtidos são apresentados na Figura 8.1.

Semente nº 1	Pacotes	Pacotes/segundo	Bytes	Bytes/segundo
Sem ataque				
Interface 0	499	6,011	126046	1518,384
Interface 1	-	-	-	-
Interface 2	-	-	-	-
Com ataque				
Interface 0	419	4,936	135518	1596,351
Interface 1	196	2,479	99884	1263,510
Interface 2	171	2,115	100638	1244,815
Semente nº 2	Pacotes	Pacotes/segundo	Bytes	Bytes/segundo
Sem ataque				
Interface 0	-	-	-	-
Interface 1	373	5,157	105186	1454,274
Interface 2	-	-	-	-
Com ataque				
Interface 0	389	4,126	135518	1437,396
Interface 1	204	2,163	100871	1069,906
Interface 2	186	1,972	104867	1112,291
Semente nº 3	Pacotes	Pacotes/segundo	Bytes	Bytes/segundo
Sem ataque				
Interface 0	-	-	-	-
Interface 1	-	-	-	-
Interface 2	381	4,041	113291	1201,641
Com ataque				
Interface 0	346	3,669	135518	1437,396
Interface 1	247	2,619	100871	1069,906
Interface 2	192	2,036	104867	1112,291

Figura 8.1 – Resultado da simulação para três sementes distintas

9. Conclusão

A simulação de um ataque distribuído de negação de serviço envolve vários campos de conhecimento específicos, além do estudo aprofundado dos mesmos. Por isso, a atividade de escolha de ferramentas para o uso no projeto é complexa. Faz-se necessário o estudo de todas essas áreas para obter das ferramentas a melhor representatividade na simulação. Essa extensa revisão bibliográfica enriquece o produto final, pois toda etapa do projeto precisa ser confrontada com vários artigos da área.

A reprodução do trabalho de Souza e Júnior (2011) foi de grande valor para introduzir o simulador NS-3 e seu modo de operação. De toda forma, as simplificações acerca da topologia, do tráfego de fundo e da modelagem da *botnet* não condizem com a realidade existente nos cenários encontrados na literatura.

O simulador e o gerador de topologias escolhidos não apresentaram, ao longo das atividades realizadas, qualquer instabilidade, mostrando-se adequados ao presente trabalho. Ambos revelaram-se bastante flexíveis, possibilitando a alteração dos parâmetros desejados para a adequação às características propostas existentes na literatura.

Como foi explicado ao longo do trabalho, a escolha adequada de um modelo de topologia é uma tarefa árdua. Essa área de pesquisa permanece em constante evolução. As métricas que melhor avaliam a Internet ainda não estão definidas, e mesmo as existentes ainda são descorrelacionadas. Esforços têm sido feito no sentido de conseguir capturar dados que forneçam um panorama mais amplo sobre a Internet, algo que não existe atualmente. Dessa forma, optou-se por empregar os modelos mais referenciados na literatura existente.

A ausência de artigos que determinem alguns parâmetros da simulação fez com que os autores assumissem determinados valores para a instância da simulação realizada. Já em outros casos, como o do roteamento e o do endereçamento IP, limitações do próprio *software* empregado impossibilitaram maior sofisticação da solução adotada.

A geração de tráfego de fundo é bem modelada para filas em servidores. O PPBP simula múltiplas fontes de tráfego oriundas do mesmo nó, existem validações

para o modelo para o caso SSQ. No entanto, os valores dos parâmetros do PPBP adequados à simulação desejada não foram encontrados na literatura.

A ferramenta produzida por este projeto buscou o máximo de validação por meio do emprego de modelos já consagrados no meio científico. Paralelamente, buscou-se oferecer flexibilidade de configuração da simulação para o usuário. Ainda resta validar a integração de todos os modelos empregados, através da coleta e análise de informações produzidas pela ferramenta. Contudo, o projeto pode ser considerado um sucesso pela abrangência dos assuntos pertinentes abordados, tratados com o máximo rigor possível a fim de produzir uma ferramenta útil para pesquisas na área.

ANEXO A – GLOSSÁRIO

Malware	*Malware* é uma abreviação de *malicious code* (código malicioso) e significa um coletivo para qualquer software malicioso que entra em um sistema sem autorização de seu usuário. (VINOD e GAUR, 2009)
Sistema autônomo	É um conjunto conectado de um ou mais prefixos de roteamento IP administrados por um ou mais operadores de rede com uma única e claramente definida política de roteamento (HAWKINSON, 1996).
Trace –	*Trace* é um arquivo de registro do comportamento da rede em determinado período. O arquivo de *trace* geralmente possui informações sobre quantidade de pacotes trafegados pela rede, disponibilidade de servidores, destino e origem dos pacotes, entre outros.
GNU GPL	É uma licença de *software* livre que, fundamentalmente, se baseia em quatro liberdades, permitindo a distribuição e reaproveitando dos programas, mas mantendo os direitos do autor.

ANEXO B – EXECUÇÃO DE TRABALHOS ANTERIORES

Com a finalidade de realizar uma ambientação para esse trabalho, foi estudado e reproduzido o projeto de Souza e Júnior. Ele trata de uma simulação de ataque distribuído de negação de serviço executado com a topologia da Figura B.0.1.

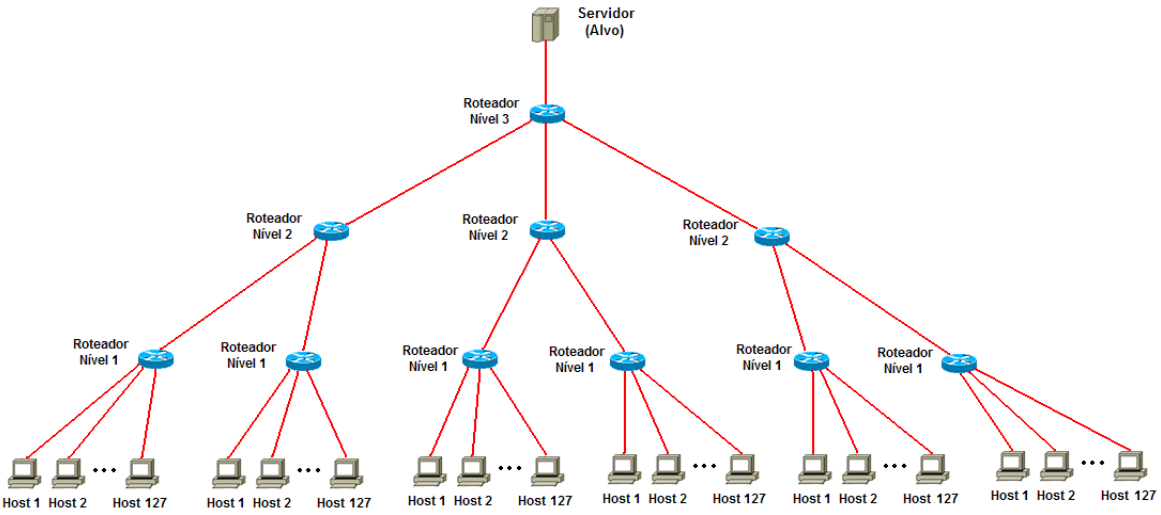

Figura B.0.1 - Topologia de simulação de um ataque de negação de serviço (SOUZA, 2011).

Os autores definem como subrede os *hosts* ligados aos roteadores de nível 1. Cada subrede possui 127 hosts, sendo 40% destes aleatoriamente escolhidos para realizar o ataque. As larguras de banda dos enlaces das subredes são definidas segundo a proporção real da Rede Rio. Para os níveis subseqüentes, a largura de banda é, respectivamente, 100 Mbps e 1Gbps. Entre o roteador de nível 3 e o servidor, 100 Mbps.

Os hosts não atacantes implementam um tráfego de fundo que segue a distribuição de Poisson para o número de pacotes e a de Pareto para o tamanho dos pacotes. Os nós atacantes realizam *TCP Flood*, *UDP Flood* e *ICMP Flood*. Todos os hosts da simulação estão sujeitos a uma probabilidade de disponibilidade que visa simular o tempo que o host fica ligado por dia. Foram realizadas diversas simulações e seus resultados foram gravados em formato PCAP.

A implementação dessa simulação foi feita no NS-3 de versão 3.9, havendo diferenças para a 3.13, empregada nesse trabalho. Além dos erros de referência, algumas classes do NS-3 tiveram seus cabeçalhos alterados. Assim como foi feito na simulação original, foi necessária a criação de um novo módulo (chamado “pfc”) para o NS-3 contendo as classes utilizadas na simulação. O código também foi alterado para tornar a visualização possível, como visto na Figura B.0.2. Houve necessidade de instanciar um objeto *CommandLine* para receber parâmetros da simulação através da linha de comando.

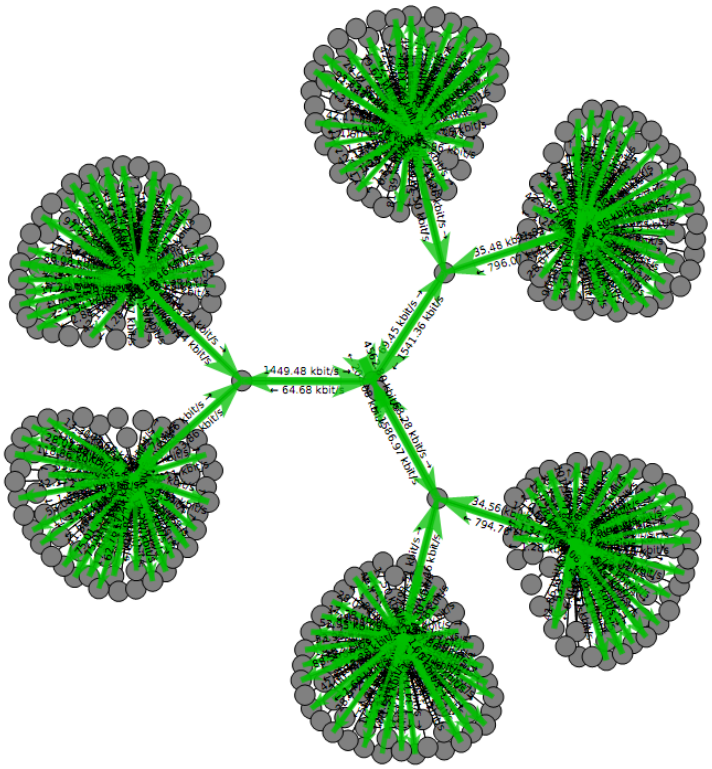

Figura B.0.2 – Visualização da Simulação de Sousa e Júnior.

O manual do NS-3 descreve o processo de criação de módulos, porém os erros resultantes da diferença de versão não foram sanados. A solução para o problema foi encontrada por meio da revisão de todos os *links* contidos na simulação, comparando-os com a documentação de referência do código fonte do NS-3. Após um estudo detalhado do *script* dos autores e das classes do NS-3, os erros foram resolvidos, e foi possível reproduzir os resultados obtidos.

ANEXO C – USO DA FERRAMENTA

Para criar uma instância do ambiente de simulação, deve-se criar um script de simulação e declarar um objeto da classe *PFCHelper*. Este deverá ser configurado conforme as funções descritas no Anexo D. Após isto, a execução da simulação propriamente dita se inicia com a chamada do comando *Run* do objeto instanciado.

A simulação é medida em segundos e se inicia no tempo zero. O seu tempo de término é definido pelo usuário. Em seus parâmetros de configuração, pode-se optar pela geração do arquivo PCAP, contendo o *trace* do tráfego recebido e enviado pelo nó alvo.

Pode-se, ainda, acompanhar a visualização da simulação em tempo real. Para isto, basta adicionar o parâmetro "--vis" ao fim da chamada do *script* de simulação. Todavia, para o tamanho de topologia que se pretende simular com a ferramenta desenvolvida, esta opção não é aconselhável devido ao grande consumo de recursos computacionais. A Figura C.1 ilustra um exemplo de *script* de simulação.

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*-
*/

#include <string>
#include "ns3/brite-module.h"

using namespace ns3;

int main (int argc, char *argv[])
{
  CommandLine cmd;
  cmd.Parse (argc,argv);

  PFCHelper* pfchelper = new PFCHelper();

  pfchelper->Run();

  return 0;
}
```

Figura C.0.1 – Exemplo de *script* de simulação empregando a ferramenta desenvolvida

ANEXO D – INTERFACE DA FERRAMENTA DE SIMULAÇÃO

Função	**Valor Padrão**
`PFCHelper(void)`	-
Descrição	
Construtor. Carrega os valores padrão dos parâmetros da simulação.	

Função	**Valor Padrão**
`void Run(void)`	-
Descrição	
Executa a simulação. Para que a simulação ocorra, todos os parâmetros devem ter sido configurados corretamente.	

Função	**Valor Padrão**
`void SetAttackDuration(double)`	50
Descrição	
Define a duração do ataque. Esse valor deve estar coerente com o início do ataque e a duração da simulação.	

Função	**Valor Padrão**
`void SetAttackPort(unsigned int)`	28
Descrição	
Define porta do ataque UDP. É importante que essa porta conste nas portas UDP definidas em SetUdpPorts.	

Função	**Valor Padrão**
`void SetBriteConfigurationFile(std::string)`	-
Descrição	
Informa o caminho absoluto para o arquivo de configuração do BRITE.	

Função	**Valor Padrão**
`void SetBriteSeedFiles(std::string, std::string)`	-
Descrição	
Informa os caminhos absolutos para o arquivo de *seed* que será utilizado pelo BRITE e para o novo arquivo de *seed* a ser gerado.	

Função	Valor Padrão
`void SetHostAvailability(double[])`	{0.08, 0.25, 1.0}
Descrição	
Define os valores de disponibilidade para os nós. Esses valores devem ser porcentagens, representando a fração do tempo ativo de um nó durante a simulação. Esse parâmetro deve ser coerente com o quantidade de probabilidades definidas em SetHostAvailabilityProbability. Os valores devem estar entre 0 e 1.	

Função	Valor Padrão
`void SetHostAvailabilityProbability(double[])`	{0.3, 0.6, 0.1}
Descrição	
Define a probabilidade de cada entrada dos valores de disponibilidade para os nós. Esses valores representam a probabilidade que o tempo de disponibilidade de mesmo índice, definido em SetHostAvailability, ocorrerá. Os valores devem estar entre 0 e 1, somando 1 no total. Esse parâmetro deve ser coerente com o quantidade de probabilidades definidas em SetHostAvailability.	

Função	Valor Padrão
`void SetIcmpAttack(bool)`	True
Descrição	
Define se haverá ataque ICMP.	

Função	Valor Padrão
`void SetInfectedPercentage(double)`	0.25
Descrição	
Define a porcentagem de nós folhas infectados. Esse valor deve estar entre 0 e 1.	

Função	Valor Padrão
`void SetMaximumApplicationsPerNode(int)`	5
Descrição	
Define o número máximo de aplicações por nó folha.	

Função	Valor Padrão
`void SetMaxUdpPacketsToSend(int)`	(1<<15) - 1
Descrição	
Define o número máximo de pacotes a serem enviados pela aplicação de ataque UDP (*UDPFlood*).	

Função	Valor Padrão
`void SetMinimumApplicationsPerNode(int)`	1
Descrição	
Define o número mínimo de aplicações de tráfego de fundo por nó folha.	

Função	Valor Padrão
`void SetPcap(bool)`	True
Descrição	
Define se o arquivo PCAP será gerado para o nó alvo.	

Função	Valor Padrão
`void SetPpbpBurstIntensity(std::string)`	1Mbps
Descrição	
Define a *bit-rate* das rajadas PPBP.	

Função	Valor Padrão
`void SetPpbpMeanBurstArrivals(double)`	10
Descrição	
Define o número médio de rajadas ativas do PPBP.	

Função	Valor Padrão
`void SetPpbpMeanBurstTimeLength(double)`	0.2
Descrição	
Define a média da distribuição de Pareto do PPBP.	

Função	Valor Padrão
`void SetPpbpPacketSize(int)`	100
Descrição	
Define o tamanho em *bytes* dos pacotes gerados pelo PPBP.	

Função	Valor Padrão
`void SetSimulationDuration(double)`	100
Descrição	
Define a duração em segundos da simulação. A simulação ocorrerá do tempo 0 até o valor atribuído por essa função.	

Função	Valor Padrão
`void SetStartAttackTime(double)`	10
Descrição	
Define o início, em segundos, do ataque. Essa variável deve estar consistente com o intervalo de simulação.	

Função	Valor Padrão
`void SetTargetId(unsigned int)`	0
Descrição	
Define qual dos nós folhas é o alvo do ataque. O índice do alvo é numerado de zero até $n - 1$, sendo que n é a quantidade de nós folhas.	

Função	Valor Padrão
`void SetTcpPorts(unsigned int[])`	{443, 80, 1935, 25, 21, 110, 143, 23, 22, 1000}
Descrição	
Define as portas TCP que serão utilizadas pelas aplicações durante a simulação. O usuário pode definir quantas portas quiser, desde que a quantidade de portas seja compatível com o vetor de probabilidades definido em SetTCPPortsProbability.	

Função	Valor Padrão
`void SetTcpPortsProbability(double[])`	{0.435, 0.338, 0.0024, 0.0044,0.001, 0.0002, 0.0002, 0.0032, 0.015, 0.2006}
Descrição	
Define as probabilidades de ocorrência de comunicação nas portas de mesmo índice definidas em SetTCPPorts. Seus valores devem estar entre 0 e 1. A soma desses valores deve ser 1.	

Função	Valor Padrão
`void SetTcpProbability(double)`	0.812
Descrição	
Define a probabilidade de ocorrência de uma comunicação TCP. Para maior coerência da simulação, é recomendável que esse valor seja complementar ao valor atribuído em SetUDPProbability, pois não são simuladas aplicações em outros protocolos no tráfego de fundo.	

Função	Valor Padrão
`void SetUdpAttack(bool)`	True
Descrição	
Define se haverá ataque UDP.	

Função	Valor Padrão
`void SetUdpPorts(unsigned int[])`	{53, 12488, 32064, 36457, 42552, 32222, 2000}
Descrição	
Define as portas UDP que serão utilizadas pelas aplicações durante a simulação. O usuário pode definir quantas portas quiser, desde que a quantidade de portas seja compatível com o vetor de probabilidades definido em SetUDPPortsProbability.	

Função	Valor Padrão
`void SetUdpPortsProbability(double[])`	{0.15, 0.001,0.001,0.001,0.001, 0.001, 0.8495}
Descrição	
Define as probabilidades de ocorrência de comunicação nas portas de mesmo índice definidas em SetUDPPorts. Seus valores devem estar entre 0 e 1. A soma desses valores deve ser 1.	

Função	Valor Padrão
`void SetUdpProbability(double)`	0.188
Descrição	
Define a probabilidade de ocorrência de uma comunicação UDP. Para maior coerência da simulação, é recomendável que esse valor seja complementar ao valor atribuído em SetTCPProbability, pois não são simuladas aplicações em outros protocolos no tráfego de fundo.	

Função	Valor Padrão
`void SetVisualize(bool)`	True
Descrição	
Define se serão utilizadas as coordenadas cartesianas oferecidas pelo BRITE para o posicionamento dos nós.	

BIBLIOGRAFIA

PENG, Tao; LECKIE, Christopher; RAMAMOHANARAO, Kotagiri.**Survey of network-based defense mechanisms countering the DoS and DDoS problems**.Department of Computer Science and Software Engineering.University Of Melbourne.Australia, 2007.

British Standard 7799.**Information Security Management Systems – Specification with guidance for use**. British Standard, 2002.

ISO/IEC 27000.**Information technology – Security techniques – Information security management systems – Overview and vocabulary**. International Organazation for Standardization, 2005.

FOURUZAN, Behrouz. **Data Communications and Networking**. Quarta edição. Bookman, 2009.

TANEMBAUM, A. S.. **Computer Networks**. Quarta edição. Campus, 2005.

GLIGOR,V. D. 1984. **A note on denial-of-service in operating systems.** IEEE Trans. Softw. Eng.

TANGMUNARUNKIT, Hongsuda, **"Network Topology Generators:Degree-Based vs. Structural,"**. 2002.

MOORE, Andrew; MORTIER, Richard. **Network Topologies: Inference Modeling, and Generation.** 2008.

ALBERT, Réka; BARABÁSI, Albert-László. **Topology of evolving networks: local events and universality**. Physical Review Letters, 2000.

BU,Tian; TOWSLEY, Donald F. **On Distinguishing between Internet Power Law Topology Generators.** INFOCOM 2002.

FALOUTSOS, Michalis; FALOUTSOS,Petros; FALOUTSOS, Christos. **On power-law relationships of the Internet topology**. SIGCOMM '99.

CHEN, Qian; CHANG, Hyunseok; GOVINDAN, Ramesh; JAMIN, Sugih; SHENKER, Scott J.; WILLINGER,Walter. **The Origin of Power Laws in Internet Topologies Revisited**. IEEE INFOCOM 2002, 2001.

MEDINA, Alberto; LAKHINA,Anukool; MATTA, Ibrahim, BYERS, John W. **BRITE: Na Approach to Universal Topology Generation**. IEEE MASCOTS, 2001.

WINICK, Jared; JAMIN, Sugih. **Inet-3.0: Internet Topology Generator**. Relatório Técnico 456-02, EECS, University of Michigan, 2002.

CAIDA: The Cooperative Association for Internet Data Analysis. Acesso em 01/03/2012. Disponível em : http://www.caida.org

NS-3 PROJECT. **NS-3 Tutorial Realease ns-3-dev.** 2012 Acesso em 10/03/2012. Disponível em: http://www.nsnam.org/docs/tutorial-pt-br/ns-3-tutorial.pdf

NS-3: Network Simulator. Acesso em 01/03/2012. Disponível em: http://www.nsnam.org/

BRITE: Boston University Topology Representative Generator. Acesso em 01/03/2012. Disponível em: http://www.cs.bu.edu/brite/

SCALZO, Frank. **Recent DNS Reflector Attacks**. VeriSign, 2006.

ASSADHAN, Basil; MOURA, José M. F. **Detecting Botnets using Command and Control Traffic**. IEEE, 2009.

JUNIOR, Fábio Luiz ; SOUSA, Vinícius Hessel Benedito de. **Implementação de um ataque de negação de serviço no ambiente Network Simulator**. Rio de Janeiro: Instituto Militar de Engenharia, 2011.

PROLEXIC ATTACK REPORT: Q1 2012. Prolexic Technologies, 2012. Quadrimestral.

UHLIG, Steve. **Tuning Topology Generators Using Spectral Distributions.** 2008.

ZUKERMAN, Moshe. **Introduction to Queueing Theory and Stochastic Teletraffic Models**. University of Melbourne, 2012.

ZUKERMAN, Moshe. **Internet Traffic Modeling and Future Technology Implications**. University of Melbourne, 2003.

AMMAR, Doreid; BEGIN, Thomas. **A New Tool for Generating Realistic Internet Traffic in NS-3**. Université Lyon, 2011.

RESNICK, Sidney I. **Heavy-tail Phenomena: Probabilistic and Stastistical Modeling**. Springer, 2007.

KARAGIANNIS, Thomas; MOLLE, Mart; FALOUTSOS, Michalis. **Long-Range Dependence: Ten Yers of Internet Traffic Modeling**. University of California, 2004.

HAWKINSON, J.; BATES, T. **RFC 1930: Guidelines for creation, selection, and registration of an Autonomous System (AS)**. 1996.

MAWI Working Group: Traffic Archive. Acesso em 26/04/2012. Disponível em : http://mawi.wide.ad.jp/mawi/

CHAKRABARTI, Deepayan; FALOUTSOS, Christos. **Graph Mining: Laws, Generators, and Algorithms.** Carnegie Mellon University, 2006.

LU, Qindong; HUANG, Xinli; WANG, Yu. **Leetar: a new hierarchical-structural topology generator for the Internet**. IEEE, 2011.

HERNANDEZ, Javier Martin; KLEIBERG, Tom. **A Qualitative Comparsion of Power Law Generators**. Delft University of Technology, 2006.

MAGONI, Damien; PANSIOT, Jean-Jacques. **Analysis and Comparsion of Internet Topology Generators**. Université Louis Pasteur, 2001.

MACIEL, Joylan N. **Um novo gerador de topologias da Internet**. Universidade Federal do Paraná, 2008.

MANSFIELD, Steve. **Hacktivism: assessing the damage**. Elsevier, 2011.

RILEY, George F. **Using the Georgia Tech Network Simulator.** 2008. Acesso em 10/03/2012. Disponível em: http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/docs/GTNetS_manual.pdf

VARGA, András. **OMNeT++ User Manual**. 2011. Acesso em 10/03/2012. Disponível em: http://omnetpp.org/doc/omnetpp/Manual.pdf

DAGON, David. **A Taxonomy of Botnet Structures**. Georgia Institute of Technology, 2007.

Boeing Company. **Common Open Research Emulator.** 2012. Acesso em 10/03/2012. Disponível em: http://pf.itd.nrl.navy.mil/core/core_manual.pdf

KUKEC, Ana; MIKUC Miljenko; VASIC; Valter; ZEC; Marko. **IMUNES Manual.** 2011. Acesso em: 10/03/2012. Disponível em: http://imunes.tel.fer.hr/imunes/dl/imunes_ug_20110907.pdf

MESSMER, Ellen. **America's 10 most wanted botnets.** 2009. Acesso em 10/03/2012. Disponível em: http://www.networkworld.com/news/2009/072209-botnets.html

VINOD, P.; GAUR, V. Laxmi, M. S. **Survey on Malware Detection Methods**. 2009. Acesso em 10/03/2012. Disponível em: http://www.security.iitk.ac.in/hack.in/2009/repository/proceedings_hack.in.pdf#page=82