DEFINIÇÃO DE ASSINATURAS DE ATAQUE EM STATL[1]

Mário Cléber Bidóia[2] (Ciência da Computação/UNIVEM)
Orientadora: Kalinka Regina L. Jaquie Castelo Branco[3]

Resumo

Atualmente, muitas organizações conectadas à Internet tiveram um considerável aumento no orçamento para tratar de forma mais eficaz e eficiente a segurança de sua rede. Ataques são freqüentemente executados a *hosts* e redes de computadores, sendo alguns destes: vírus, worms, cavalos de tróia, *port scanning* - varredura de portas, DoS (*Denial of Service*) – negação de serviço, DDoS (*Distributed Denial of Service*) – negação de serviço distribuída, *spoofing* DNS (*Domain Name System*). Uma das formas efetivas de defesa contra os ataques é a utilização de IDS (*Intrusion Detection System*) – Sistema de Detecção de Intrusão. Uma vez que os Sistemas de Detecção de Intrusão são necessários, assinaturas de ataques para facilitar essas detecções também se fazem necessárias. Dessa maneira, confeccionar cenários de ataques para esses sistemas constitui tarefa importante no contexto de redes de computadores. Atualmente existem vários IDS, um que tem se destacado é o STAT – *State Transition Analysis Technique* por ter código aberto e permitir o uso de uma metodologia para descrever penetrações de computadores por meio de cenários de ataques. Estes são representados por uma sucessão de transições que caracteriza a evolução do estado de segurança do sistema. Complementarmente possui o *framework* STAT, que provê uma linguagem para especificação de cenários de ataques, a STATL. Uma especificação de STATL é a descrição de um cenário de ataque completo. O ataque é modelado como uma sucessão de transições de estados que traz um sistema de um estado inicial "seguro" para um estado final "assumido compromisso". Este trabalho se dá a partir do uso da ferramenta STAT efetuando-se a composição de diferentes cenários e da geração de algumas assinaturas. A ferramenta é de instalação fácil e provê recursos eficientes para a geração de cenários. O entendimento de máquinas de estado se faz necessário para que tanto a definição quanto a especificação dos diagramas de transição sejam efetuados de forma correta e possam, com isso, gerar resultados coesos.

Palavras-chave: 1. Segurança de computadores 2. Sistemas de Detecção de Intrusão (IDS) 3. Software livre

1 Artigo resultado de Iniciação Científica realizada no ano de 2008 (UNIVEM - Marília/SP).

2 Bacharel em Ciência da Computação (UNIVEM - Marília/SP). E-mail: mariobidoia@hotmail.com

3 Docente do Instituto de Ciências Matemáticas e de Computação - ICMC (São Carlos/SP).

Abstract

DEFINITION OF ATTACK SIGNATURES BY STATL

Currently, many organizations connected to the Internet had a significant increase in the budget to deal more effectively and efficiently the security of your network. Attacks are frequently performed to hosts and networks of computers, some of those viruses, worms, trojans, port scanning, DoS (Denial of Service) - DoS, DDoS (Distributed Denial of Service) - Disclaimer distributed service, spoofing DNS (Domain Name System). One of the ways for effective defense against attacks is the use of IDS (Intrusion Detection System). Once Systems Intrusion Detection are needed, attack signatures to facilitate these probes are also necessary. This way, cook attack scenarios for these systems is important task in the context of computer networks. Currently there are several IDS, one that has stood the STAT - State Transition Analysis Technique for having open source and allow the use of a methodology to describe computer penetrations through attack scenarios. These are represented by a succession of transitions that characterizes the evolution of system security. In addition has the STAT framework that provides a language for specifying attack scenarios, the STATL. A specification of STATL and the description of an attack scenario complete. The attack is modeled as a succession of state transitions that brings a system from an initial state "safe" for a final state "assumed commitment." This work takes place from the use of the tool STAT effecting the composition of different scenarios and the generation of some signatures. The tool is easy to install and provides powerful features for generating scenarios. The understanding of state machines is necessary for both definition and specification of transition diagrams are made correctly and can thereby generate cohesive results.

Key-words: 1. Computer Security 2. Systems Intrusion Detection (IDS) 3. Open Source

INTRODUÇÃO

Atualmente, muitas organizações conectadas à Internet tiveram um considerável aumento no orçamento para tratar de forma mais eficaz e eficiente a segurança de sua rede (Xiang, Zhou e Li, 2006). Ataques são freqüentemente executados a *hosts* e redes de computadores, sendo alguns destes: vírus, *worms*, cavalos de tróia, *port scanning* - varredura de portas, DoS (*Denial of Service*) – negação de serviço, DDoS (*Distributed Denial of Service*) – negação de serviço distribuída, *spoofing* DNS (*Domain Name System*).

Uma das formas efetivas de defesa contra os ataques é a utilização de IDS (*Intrusion Detection System*) - Sistema de Detecção de Intrusão. Segundo Balasubramaniyan (1998), detecção de intrusão pode ser definida como sendo o ato de revelar qualquer conjunto de ações que comprometa a integridade, confidencialidade ou a disponibilidade do sistema, identificando agentes que estejam utilizando um sistema computacional sem autorização, e reconhecendo pessoas que têm acesso legítimo ao sistema, mas estão abusando de seus privilégios.

Uma vez que os Sistemas de Detecção de Intrusão são necessários, assinaturas de ataques (Northcutt, 2002) para facilitar essas detecções também são necessárias. Dessa maneira, confeccionar cenários de ataques para esses sistemas constitui tarefa importante no contexto de redes de computadores.

Existem atualmente vários IDS; um que tem se destacado é o STAT – *State Transition Analysis Technique* – (Vigna, Valeur e Kemmerer,2003). O STAT é uma metodologia para descrever penetrações de computadores por meio de cenários de ataques. Estes são representados por uma sucessão de transições que caracteriza a evolução do estado de segurança do sistema.

O *framework* STAT provê de uma linguagem para especificação de cenários de ataques, a STATL (Vigna, Valeur e Kemmerer,2003).

Uma especificação de STATL é a descrição de um cenário de ataque completo. O ataque é modelado como uma sucessão de transições de estados que traz um sistema de um estado inicial "seguro" para um estado final "assumido compromisso". Estados são usados para caracterizar instantes diferentes de um sistema durante a evolução de um ataque.

O presente trabalho objetivou confeccionar cenários de ataque no STATL para geração de assinaturas.

I TIPOS DE ATAQUES

Um ataque é uma tentativa de acesso não autorizado, ou seja, é o que acontece quando uma ameaça[4] tenta levar vantagem sobre as vulnerabilidades[5] de um determinado sistema. Um ataque (ou intrusão) pode ser formalmente definido, segundo Heady (1990) como: "qualquer conjunto de ações que tentem comprometer a integridade, confidencialidade ou disponibilidade dos dados e/ou do sistema".

Pode-se ainda definir um ataque, de forma mais abrangente, como qualquer violação à política de segurança de um sistema.

Os ataques podem ser divididos em duas categorias: ataques ativos e ataques passivos (Kurose e Ross, 2005). Ataques ativos correspondem às ameaças que alteram o sistema na tentativa de levarem vantagem sobre uma vulnerabilidade do mesmo, enquanto que os ataques passivos consistem em ameaças que simplesmente observam informações que trafegam no sistema, contrariamente, não são inseridas informações que explorem as vulnerabilidades desse sistema.

Segundo Kurose e Ross (2005), os principais tipos de ataques à rede de computadores são: vírus, *worms*, *port scanning* – varredura de portas, DoS (*Denial of Service*) – negação de serviço, DDoS (*Distributed Denial of Service*) – negação de serviço distribuído. Esses ataques são descritos nas seções que seguem.

4 Ameaça a um sistema, neste texto, consiste no querer atentar contra a segurança desse sistema.

5 Vulnerabilidade pode ser entendida como sendo o ponto frágil de um projeto, implementação ou configuração de sistema, que permite exploração com intenções danosas.

1.1 Vírus, *Worms* e Cavalos de Tróia

Vírus, *worms* (vermes) e cavalos de tróia (*trojans horses*), típicos exemplos de códigos maliciosos, são considerados os maiores problemas dos administradores de redes.

Os vírus são programas de computadores (ou fragmentos de programas), geralmente maliciosos, que se propagam infectando o computador, gerando cópias de si mesmo e tornando-se parte de outros programas de computador. Eles dependem da execução do programa hospedeiro para serem ativados e continuarem o processo de infecção (Cert.br, 2006). Além de serem capazes de se reproduzirem, eles podem também corromper arquivos e sistemas.

A propagação dos vírus se dá através de disquetes, CD-ROMS, documentos infectados que são executados, de *e-mails* (com anexos infectados), de programas piratas, da execução de *downloads* de procedência duvidosa (*Orkut, Msn, Skype*), entre outros. Em todos esses exemplos, faz-se necessária a presença e atuação do ser humano (Moura *et al.*, 1999).

Alguns vírus são pré-programados para danificar o computador de forma a corromper os programas, excluir arquivos ou até mesmo simplesmente construídos para transmitir mensagens cujo intuito é chamar a atenção de um número grande de pessoas. Independentemente do intuito, constituem, normalmente, situações que acabam por culminar na queda ou parada temporária do sistema.

Os *worms* podem, por outro lado, ser considerados uma espécie de vírus, que, devido a sua forma de reprodução, são considerados mais desastrosos que os vírus normais (MOURA, 1999).

Por definição, os *worms* são programas capazes de se reproduzirem de um computador para o outro, assim como os vírus, mas, diferentemente dos vírus, os *worms* não utilizam um programa como hospedeiro. Eles possuem a capacidade de se propagarem automaticamente no próprio computador ou de computador para computador, realizando, assim, a infecção (Cert.br 2006).

Um exemplo de propagação de *worms* é o do *Melissa* (CERT, 1999), que, após infectar um computador, ele mesmo procura os endereços no programa cliente de *e-mail* da vítima e transmite, de forma transparente para o usuário, o documento infectado para todos os *e-mails* cadastrados no contato da vítima.

Os *worms* são considerados mais perigosos que os vírus comuns, pois, além da não necessidade da intervenção humana, eles fazem uso de situações normais de operação do sistema (por parte do usuário) para efetuarem sua propagação.

Os *trojans*, ou mais conhecidos como cavalos de tróia, são um exemplo de código malicioso que aparentemente se mostra inofensivo. Eles vêem escondidos em cartões virtuais, protetores de tela, fotos. Esses programas, além de executarem funções para as quais foram aparentemente projetados, também executam outras funções normalmente maliciosas e sem o conhecimento do usuário (Cert.br 2006). Esse código malicioso costuma executar automaticamente diversas funções como captura de dados, alterações de características e configurações de um sistema ou permitir acesso remoto em uma base cliente/servidor.

Os métodos utilizados para a instalação desses programas no sistema são muito variados, principalmente porque os programas estão tipicamente escondidos no meio do código de outro programa ou são inseridos por meio de acesso não autorizado.

De um modo geral, as formas de se combater os vírus, *worms* e cavalos de tróia são:

- utilizar (e manter atualizado) antivírus para detectar a presença dos mesmos;
- não abrir *e-mail*s de procedência duvidosa;
- não efetuar *download* de *softwares* e documentos de sites duvidosos; e
- estar atento sempre para as novas ameaças.

Uma vez que as ameaças, atualmente, são produzidas de formas variadas, fazendo uso de técnicas cada vez mais sofisticadas e audaciosas, existe a preocupação em se prover técnicas mais eficientes no combate aos diferentes tipos de ataques.

1.2 *Port Scanning* – Varredura de Portas

Port scanning é uma técnica de varredura de portas TCP (*Transmission Control Protocol*) comum a *crackers* para reconhecimento de sistemas-alvo. Esse ataque consiste em testar as portas de um *host*, ou mesmo de um grupo de *hosts*, a fim de determinar quais dessas portas estão em condições de aceitar conexões. Dessa forma, um programa de *port scan* é aquele que averigua conexões cujos números de porta são bem conhecidos para detectar informações e serviços em execução no sistema-alvo. Assim, com base na lista de portas que estão aguardando conexões, o *cracker* pode escolher entre um ou outro método de invasão (Kurose e Ross, 2005).

O *port scanning*, eventualmente, também é capaz de revelar outras informações de interesse do atacante, tais como o sistema operacional em execução, a partir da análise de como o alvo reage aos eventos gerados durante a varredura.

Podem ocorrer ligeiras diferenças na forma como é realizado o *port scanning*, mas ele consiste, basicamente, em enviar uma série de requisições, seja via TCP ou UDP (*User Datagram Protocol*), para um *range* de portas de um determinado *host* e verificar as respostas. Essas respostas podem ser utilizadas para determinar quais serviços estão ativos, por exemplo: *Web* (*HTTP - Hiper Text Transfer Protocol* – porta 80), transferência remota de arquivos (*FTP - File Transfer Protocol* – porta 21), ou *e-mail* (*SMTP – Simple Mail Tranfer Protocol* – porta 25) (Kurose e Ross, 2005).

Dentre os diferentes tipos de varredura, podem-se destacar:

Varredura padrão: a varredura é iniciada com o envio de um pacote TCP SYN (*Synchronize*) para o alvo. O *three-way handshake* é executado de forma completa e a conexão entre o *host* que executa o *port scanner* e o alvo é estabelecida, possibilitando a obtenção de informações mais abrangentes. Entretanto, facilita que a varredura seja percebida e sua origem detectada.

Varredura TCP SYN: a varredura também é iniciada com o envio de um pacote TCP SYN para o alvo, entretanto a conexão não é completada. Uma vez que o alvo responde à solicitação de abertura de conexão (por meio de um pacote TCP, ACK (*Acknowledgement*) ou RST (*Reset*)), o *port scanner* já é capaz de avaliar se a porta está em estado de escuta e interrompe o processo de conexão. Por esse motivo, esse tipo de varredura é muitas vezes conhecido também como varredura semi-aberta. A vantagem desse método é que ele dificulta a detecção da origem da varredura.

Varredura *Stealth Scanning:* a varredura é, nesse caso, iniciada com um pacote que simula uma conexão já existente, e não mais com um pacote TCP SYN, e, desse modo, a forma de resposta ao pacote revela não só o estado da porta, mas consegue fornecer informações extras sobre o estado do alvo. O propósito dessa varredura é evitar que filtros de pacote (bloqueio de pacotes TCP SYN) inviabilizem a varredura.

Independentemente do tipo de varredura, de um modo geral, elas são consideradas técnicas de ataque que se aproveitam do fato de que o protocolo TCP/IP (*Internet Protocol*) gerencia as conexões de forma automatizada e com um nível de crítica muito simplista. Sendo assim, um *host* responde a toda solicitação de abertura de conexão endereçada a suas portas, sem avaliar a origem do pedido (se esta é confiável ou não).

Além disso, o uso de portas conhecidas associadas a serviços padrão constitui outra característica explorada por esse tipo de ataque. Um ataque desse tipo é feito pelo uso de *softwares* específicos capazes de enviar pacotes com destino às portas do *host* alvo, monitorar respostas emitidas pelo alvo e gerar relatórios de análise.

1.3 DoS (*Denial of Service*) – Negação de Serviço

Outra categoria de ataque que constitui uma ameaça à segurança das redes de computadores é o classificado ataque de negação de serviço.

Esta categoria de ataque tem como finalidade, como sugere o próprio nome, tornar impossível a utilização dos recursos de uma rede ou de um determinado

host. Normalmente, esse tipo de ataque sobrecarrega a infra-estrutura sob ataque, ficando esta impossibilitada de realizar os trabalhos legítimos (Kurose e Ross, 2005). Desse modo, esse constitui um ataque baseado na sobrecarga da capacidade ou em uma falha não prevista.

Esse tipo de ataque tem como objetivo esgotar os recursos do sistema-alvo, forçando uma interrupção total ou parcial dos serviços. A capacidade de processamento, de armazenamento de dados e a largura de banda são alguns dos recursos visados pelas técnicas de negação de serviços.

O problema principal está focado no protocolo IP, que é altamente vulnerável a ataques DoS. Além disso, muitas ferramentas de ataques estão disponíveis para o acesso público e são relativamente fáceis de utilizar.

Embora existam técnicas de negação de serviços destinadas a atacar *hosts* e computadores pessoais, a maioria dos ataques costuma ser direcionada contra sistemas de maior porte, os quais oferecem serviços a um grande número de usuários, como é o caso de servidores *WEB*, ou que desempenham funções críticas para o funcionamento de redes e sistemas distribuídos.

Outros motivos para existirem esse tipo de falha nos sistemas é um erro básico de programadores, falhas na implementação e *bugs* (erros), além de outras peculiaridades dos sistemas operacionais, na medida em que podem oferecer oportunidades para comprometer o funcionamento do sistema (pela incapacidade de tratar erros). Desse modo, o invasor inicia da premissa de que erros existem e que deve ficar efetuando diversos tipos de testes de falhas até acontecer um erro e o sistema parar sua execução.

Esse tipo de ataque não causa perda ou roubo de informações, mas é um ataque preocupante, pois os serviços do sistema atacado ficarão indisponíveis por um tempo indeterminado. Cabe ainda salientar que, quando um computador/*site* sofre ataque DoS, ele não é invadido, mas sim sobrecarregado. Isso independentemente do sistema operacional utilizado.

Um ataque DoS pode ainda ser visto como um *worm* que se prolifera entre servidores infectados procurando novos computadores para se proliferar, entretanto, como o programa não se auto-detecta, ele se re-instala consumindo recurso das máquinas já infectadas, exaurindo assim os recursos dessas máquinas.

Ataques de DoS podem ser lançados contra roteadores de borda, *bastion hosts* e *firewalls*, sendo que roteadores, servidores DNS (*Domain Name Service*) e *firewalls* costumam ser os alvos preferenciais. Em ataques direcionados aos equipamentos de redes, o objetivo principal é tirar uma rede ou sub-rede inteira do ar e não somente algumas máquinas específicas. Existem *bugs* em vários tipos de roteadores e em outros equipamentos de conectividade que podem facilitar esses ataques. Ataques aos *firewalls* fazem com que esses percam a função de filtro, deixando passar conexões TCP para qualquer porta, sendo que um ataque DoS pode, e normalmente é, utilizado para facilitar uma invasão.

É importante ainda salientar que as técnicas de negação de serviços são freqüentemente adotadas como uma etapa intermediária de métodos de ataque mais complexos. Dessa maneira, elas servem como uma armadilha para deixar um *host* (sistema ou servidor) fora do ar a fim de que outro *host* assuma sua identidade ou, até mesmo, interrompa o funcionamento de um sistema que execute funções de segurança e controle da rede.

Existem três tipos principais de ataques de negação de serviço:

Exploração de falhas: exploram as vulnerabilidades no *software* do sistema alvo causando falhas em seu processamento ou extinguindo seus recursos.

Flooding: enviam ao sistema alvo mais informação do que ele é capaz de manipular. Mesmo que a capacidade de processamento do sistema não seja totalmente absorvida o atacante pode ser capaz de monopolizar a conexão da rede do alvo, bloqueando assim qualquer tipo de uso deste recurso.

Ataques de negativa de serviço distribuído (DDoS): possuem uma estrutura previamente montada onde diversas máquinas lançam um ataque baseado no ataque *flooding* sobre o alvo. Esses ataques fazem uso de várias máquinas, sendo essas

consideradas *zumbis*, para que o número de requisições de conexão ao servidor seja bem grande. São ataques mais complexos e eficientes, sendo que sua detecção também se torna mais complexa, chegando, algumas vezes, a se tornar até mesmo impossível.

As formas de ataque do tipo DoS mais conhecidas são:

SYN *Flooding:* constitui um ataque de inundação SYN. Neste tipo de ataque, um computador tenta estabelecer uma conexão com um servidor por meio de um sinal do TCP conhecido por SYN. Se o servidor atender ao pedido de conexão, será enviado ao computador solicitante um sinal chamado ACK. O problema é que, em ataques desse tipo, o servidor não consegue responder a todas as solicitações e, então, passa a recusar novos pedidos. Neste ataque, os pacotes TCP SYN/ACK enviados pelo alvo em resposta aos falsos pedidos de conexão não são respondidos. Isso normalmente se dá porque as solicitações são geradas com um endereço IP falso ou inválido no lugar do endereço verdadeiro da máquina que originou o ataque. Desse modo, a fila de conexões em andamento atinge seu limite configurado muito rápido e passa a descartar novas solicitações de conexão, tornando indisponíveis os serviços residentes no computador-alvo. A vulnerabilidade de um sistema a este tipo de ataque pode ser minimizada se forem adotadas configurações especiais, como, por exemplo, reduzir o limite de tempo após o qual uma conexão solicitada e não estabelecida é eliminada.

UDP *Packet Storm:* neste tipo de ataque, um computador faz solicitações constantes para que uma máquina remota envie pacotes de respostas ao solicitante. A máquina fica tão sobrecarregada que não consegue executar as funções para as quais foi previamente designada.

LAND: neste tipo de ataque, o foco são os datagramas IP. Ele baseia-se no efeito do recebimento de um datagrama IP pode ter sobre determinado sistema (os endereços de origem e destino são iguais). Isso pode produzir um *loop* que possibilita esgotar os recursos do computador, levando também a uma queda ou travamento do sistema. Pode, ainda, ocorrer variações, nas quais se tem a alteração dos campos do cabeçalho IP do pacote inválido (portas ou *bits* de controle).

Ataques baseados em ICMP (*Internet Control Message Protocol***):** este tipo de ataque aproveita-se das funcionalidades do protocolo ICMP para criar eventos capazes de afetar o funcionamento de alvos específicos. O ICMP é empregado em tarefas de controle e verificação das comunicações entre *hosts* e roteadores. Ele utiliza mensagens padronizadas que são enviadas com o propósito de checar a possibilidade de comunicar-se com um *host* de destino. As implementações padrão do TCP/IP reagirão às mensagens ICMP recebidas executando as ações apropriadas a cada caso. Elas responderão a pedidos de *ICMP_ECHO* ou poderão encerrar conexões estabelecidas a partir do recebimento de mensagens do tipo *Destination Unreachable* ou *Time to Live Exceeded*. Normalmente, não será executada qualquer modalidade de autenticação dos pedidos ou de crítica de características especiais, como repetições excessivas. Assim, uma seqüência ininterrupta de mensagens ICMP é enviada ao *host* alvo que se ocupa em responder a todas elas, consumindo desnecessariamente seus recursos. Outro ataque desse mesmo tipo, denominado *Pong,* envia mensagens ICMP a um grande número de *hosts*, endereçando-as em *broadcast*. Nas mensagens, o endereço de resposta informado é o endereço do alvo. Sendo assim, quando todos os *hosts* respondem, o alvo recebe uma grande quantidade de mensagens ICMP simultaneamente e, com isso, suas comunicações são afetadas ou interrompidas. O ataque *smurf* consiste em fazer com que *hosts* inocentes respondam requisições *echo* de pacotes ICMP para um determinado número IP falsificado pelo atacante. Constitui-se, então, em um aperfeiçoamento do ataque do tipo *Pong*, ampliando o número de *hosts* que enviarão mensagens ICMP ao alvo pelo envio da requisição falsa não apenas a um, mas a vários endereços de *broadcast.* O resultado é o envio de um grande número de pacotes ICMP de resposta ao *host* cujo número de IP foi falsificado (Kurose e Ross, 2005). Uma variante dessa técnica, denominada FRAGGLE, utiliza como protocolo de trans-

porte o UDP em lugar do TCP.

Teardrop: neste tipo de ataque, é explorado o processo de remontagem do datagrama IP fragmentado, adulterando assim informações no cabeçalho IP de forma a produzir uma situação inadequada ao processamento. Esse processo leva a falhas ou instabilidade do sistema.

Ping o'Death: neste tipo de ataque, a técnica consiste no envio ao computador alvo de um datagrama com tamanho além do limite de 65535 *bytes*. Não constitui um ataque baseado em ICMP, apesar das primeiras versões serem baseadas no *ping*. Uma vez que o datagrama ultrapassa o limite de tamanho, ele é fragmentado por não poder ser roteado, chegando à origem na forma de vários datagramas (fragmentos do datagrama original). Quando o sistema começa o processo de remontagem, acaba por gerar queda de desempenho no sistema ou a paralisação do mesmo.

Ataques de dessincronização: neste tipo de ataque, é utilizada a fragilidade existente na conexão inicial do TCP. O atacante monitora os envios de pedido de abertura de conexão e envia um sinal de RST para o *host* que foi solicitado, fechando a conexão prematuramente. Logo após esse procedimento, o atacante reenvia um sinal de solicitação de nova abertura de conexão. Desta forma, ambos os *hosts* permanecem com as portas abertas sendo utilizadas e acreditando que estão em uma conexão válida. O resultado disso é que ambos terão uma conexão dessincronizada que não pode ser utilizada para transferências de dados, mas consome recurso de ambos.

1.4 DDoS (*Distributed Denial of Service*) – negação de serviço distribuído

Ataques DDoS são problemas sérios que afetam os usuários de *Internet*, uma vez que consomem os recursos de um *host* ligado à rede que prestam serviços, tais como *e-mail* (SMTP) e páginas *Web* (HTTP).

Basicamente, os ataques DDoS são coordenados por um atacante que, de posse de *hosts* dedicados, conhecidos como *zumbis*, lança um ataque coordenado sobre uma rede ou *host* denominado vítima (Kurose e Ross, 2005). Esse tipo de ataque conquistou fama no início do ano 2000.

O DoS realiza o ataque por meio de um único computador. Com o passar dos tempos, porém, observou-se que a idéia poderia ser expandida; utilizando uma série de computadores atacantes ao mesmo tempo, poder-se-ia obter resultados mais desastrosos e eficientes. Em verdade, esse ataque potencializa os danos causados pelos ataques de negação de serviço.

Para que o ataque consiga sucesso, faz-se necessário que sejam empregados *softwares* específicos que visam a organizar esse ataque. Tem-se como exemplo o *TFN*, o *Stacheldraht* e o *Trinoo*. Essas ferramentas são instaladas em alguns *hosts* que atuarão como servidores (mestres). Paralelamente, outros *hosts* recebem também componentes de *software*, passando, por sua vez, a representar o papel de clientes (escravos).

As instalações tanto dos servidores quanto dos módulos clientes são feitas de forma não autorizada, ou seja, os componentes são embutidos em outros programas supostamente inofensivos. Ao comando do atacante, os servidores se comunicam com os clientes, determinando o início do ataque, seguidos pelos hosts que executam o módulo cliente que lançam ao mesmo tempo uma série de ataques contra o alvo ou os alvos especificados.

Dessa forma, os ataques DDoS merecem especial atenção, não apenas pela eficácia, mas também por estabelecer um novo modelo de ataque distribuído.

Tanto para efetuar como para tentar evitar um ataque DDoS fazem-se necessárias ferramentas com um alto nível de sofisticação, integrando recursos avançados que vão desde mecanismos de distribuição automatizada dos módulos clientes até comunicações criptografadas entre os servidores e os clientes. Nesse sentido, o próximo capítulo apresenta ferramentas de detecção de intrusão, denominadas IDS.

2 SISTEMAS DE DETECÇÃO DE INTRUSÃO - IDS

Segundo Balasubramaniyan (1998), detecção de intrusão pode ser definida

como sendo o ato de revelar qualquer conjunto de ações que comprometa a integridade, confidencialidade ou a disponibilidade do sistema, identificando agentes que estejam utilizando um sistema computacional sem autorização, e reconhecendo pessoas que têm acesso legítimo ao sistema, mas estão abusando de seus privilégios.

Sistemas de detecção de intrusão constituem aplicativos que monitoram a rede e são configurados para detectar qualquer comportamento que não seja considerado normal, sendo capaz de evitar um ataque, ou uma intrusão.

Como exemplo típico tem-se o *buffer overflow*, que é caracterizado pela ação do invasor em sobrescrever dados em um *buffer* não testado de um programa com seus próprios dados maliciosos. Dependendo de quais dados são sobrescritos, o programa pode parar de funcionar ou pode mudar sua execução fazendo o que o invasor deseja.

À medida que os IDS fornecem meios de inferir sobre o conteúdo das conexões permitidas de e detectar as que apresentem um comportamento suspeito ou não condizente com a política de segurança, eles tornam-se importantes (Ambrósio, 2002).

As principais características de um IDS, segundo Balasubramaniyan (1998), são:

executar continuamente sem interação humana.

ser seguro o suficiente de forma a permitir sua operação em background, entretanto não deve constituir uma caixa preta.

ser tolerante a falhas de forma a não ser afetada por uma queda do sistema (acidental ou por ações humanas).

resistir às tentativas de subversão (mudança), ou seja, deve monitorar a si próprio de forma a garantir sua segurança.

ter o mínimo de impacto no funcionamento do sistema, ou seja, gerar uma sobrecarga mínima no sistema computacional que está sendo executado.

poder detectar mudanças no funcionamento normal.

permitir fácil configuração: cada sistema possui padrões diferentes, assim o IDS deve ser adaptado de forma fácil aos diversos padrões (de acordo com a política de segurança).

adaptar-se às mudanças no sistema e ao comportamento dos usuários.

ser escalável para dar suporte à carga de monitorar várias estações e ainda produzir resultados confiáveis em tempo hábil para a tomada de decisões; e

ser difícil de ser enganado.

O último ponto faz referência aos prováveis erros que podem acontecer ao sistema. Estes podem ser classificados em: falso positivo, falso negativo e erros de subversão.

Falso positivo: ocorre quando a ferramenta trata uma ação como uma possível intrusão, quando, na verdade, trata-se de uma ação normal.

Falso negativo: ocorre quando uma intrusão real acontece, mas a ferramenta permite que ela passe como se fosse uma ação legítima.

Subversão: ocorre quando o intruso modifica a operação da ferramenta de IDS para forçar a ocorrência de falso negativo.

A maior dificuldade relativa aos sistemas de detecção de intrusão é justamente desenvolver ferramentas mais eficientes que gerem o menor número possível desses prováveis erros.

Existem diferentes classificações dos IDS. No entanto, as principais são feitas em termos da maneira como o sistema aborda o problema de detecção de intrusão e qual o método de análise utilizado para efetuar essa detecção.

2.1 Caminhos do Sistema na abordagem da Detecção de Intrusão

Segundo a forma como o problema de detecção de intrusão pode ser abordado, ele subdivide-se em:

baseado em Rede: Coletam os pacotes que trafegam na rede monitorada. Normalmente capturam dados nos pontos de estrangulamento da rede (roteadores ou *firewalls*), seja executando diretamente nesses elementos ou por meio de cópia de dados utilizando dispositivos de rede como *hubs* e *switches*. Examinam os dados que trafegam pela rede por meio de mo-

nitoração *on-line* dos pacotes, procurando por indícios de um ataque que possa estar acontecendo. Essa monitoração é realizada por de sensores espalhados pela rede que capturam todos os pacotes endereçados ao segmento de rede nos quais se encontram. Esses sensores geralmente são dispositivos passivos que têm pouco impacto no desempenho da rede. Por outro lado, esses sistemas têm dificuldade em processar pacotes em uma rede grande de tráfego intenso ou isolada por *switch* e não são capazes de analisar pacotes criptografados;

baseado em ‘‘Nós’’ da Rede: Esses sistemas constituem um tipo híbrido de IDS que conseguem superar algumas das limitações dos IDS baseados em rede. Assim como os IDS baseados em rede, esses sistemas também capturam pacotes à procura de indícios de ataques. No entanto, os sensores, ou micro-sensores, só se preocupam com os pacotes endereçados ao nó de rede no qual se encontram. Uma vez que esses sensores não capturam todos os pacotes da rede, eles podem ser mais rápidos e consumir menos recursos, causando um baixo *overhead* no sistema. Além disso, são adequados para segmentos de tráfego intenso ou isolados por *switch*; e

baseados em *Host*: Utilizam informações coletadas nos sistemas operacionais tais como registros de auditoria (*logs*), saída de programas de monitoração e informações de estado do sistema operacional, permitindo uma análise das atividades com mais confiabilidade e precisão. Essa análise determina quais processos e usuários estão envolvidos em uma invasão específica. Ao contrário dos IDS baseados em rede, esses sistemas podem saber o resultado de uma tentativa de invasão, uma vez que eles têm acesso direto e monitoram os arquivos e processos do sistema utilizados como alvos pelos invasores. Esses sistemas geralmente utilizam informações de duas fontes: auditoria do sistema operacional (mais protegido e com informações mais detalhadas) e *logs* do sistema (menores e mais claros, sendo mais fáceis de analisar). Esses sistemas são mais difíceis de gerenciar, uma vez que as informações precisam ser configuradas e gerenciadas para cada estação monitorada. Além disso, eles não podem detectar invasões direcionadas a toda a rede. Sistemas baseados em Aplicações constituem um subgrupo dos sistemas baseados em *Host* que utilizam os *logs* das aplicações para analisar os eventos ocorridos durante a execução dessas aplicações.

2.2 Quanto ao Método de Análise

A técnica de detecção de intrusão tem duas categorias principais: detecção por anomalia e detecção por abuso:

Detecções por Anomalia: Constroem padrões de uso normal dos sistemas e detectam desvios significativos desses padrões como indicativo de possíveis atividades maliciosas. Estes sistemas são, normalmente, baseados em perfis do sistema. Portanto, seu maior problema consiste nas mudanças dos perfis esperados, que tendem a gerar um grande número de falsos positivos.

Detecções por Abuso: Baseia-se na idéia de que ataques podem ser representados na forma de padrões ou assinaturas (seqüência de eventos e condições que levam a uma intrusão). Estas descrições são emparelhadas contra o fluxo de dados de auditoria que procuram evidência que o ataque modelado está acontecendo. Um dos problemas destes sistemas é que são capazes de detectar somente ataques modelados, porém eles geram um pequeno número de falsos positivos.

3 IDS EXISTENTES

Existe um grande número de ferramentas IDS, cada uma com suas características e métodos de detecção:

Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD): está sendo desenvolvida pela SRI Internacional e verifica se houve uma intrusão baseando em desvio de comportamento do usuário (anomalias) e padrões conhecidos (assinaturas). A meta principal do EMERALD é trabalhar em redes de grandes empresas (PORRAS, 1997).

Snort: um dos IDS mais populares; foi desenvolvido por (ROESCH, 2004); é um sistema de detecção de intrusão de

domínio público. É um sistema baseado em rede e utiliza uma base de assinaturas de ataques para detectar a ocorrência de intrusão.

BRO: é uma ferramenta de investigação que está sendo desenvolvida por *Lawrence Livermore National Laboratory*. Está sendo construída, em parte, para explorar as emissões relacionadas à robustez de ferramentas de IDS, isto é, avaliando quais características fazem um IDS resistir a ataques contra si mesmo (PAXSON, 1998).

NetSTAT: é a ferramenta mais recente de uma linha de ferramentas baseada em "STAT" produzida pela Universidade da Califórnia em Santa Bárbara. A atividade STAT explora o uso da análise de transição de estados para descobrir a intrusão em tempo real (VIGNA, 1998).

Como pode ser observado pelas ferramentas apresentadas anteriormente, existe uma preocupação em relação ao estudo e desenvolvimento de IDS, cada um com suas características; sejam baseados em rede ou em *host*, análise por assinaturas ou anomalias, todos procuram ser o mais perfeito possível gerando o menor número de falsos positivos (no caso de um IDS perfeito ele não geraria falsos positivos) e detectando todos os ataques ocorridos.

Neste projeto, trabalhar-se-á com o IDS STAT, que faz análises por assinaturas. O STAT foi desenvolvido por pesquisadores e estudantes da Universidade da Califórnia. Ele é um software livre (Eckmann, Vigna, Kemmerer, 2001) e pode ser baixado no site oficial do STAT <*http//www.cs.ucsb.edu/~seclab/projects/stat/ index.html*>. No site, também há trabalhos realizados sobre o STAT e sua documentação.

Como IDS deste tipo precisam ter sua base de assinaturas sempre atualizada pretende-se avaliar e especificar uma nova assinatura de ataque para o STAT. Para isso, será utilizada a STATL, que é uma linguagem de especificação criada para gerar assinaturas de ataque para o STAT.

Sendo assim, o próximo capítulo tem como objetivo apresentar mais detalhes sobre o STAT e a STATL.

4 STATE TRANSITION ANALYSIS TECHNIQUE (STAT) E STATL

A Técnica de Análise de Transição Estatal (STAT) foi concebida em 1992 como um método de detecção de intrusão por abuso para descrever penetrações de computadores como sucessões de ações que um atacante executa para comprometer a segurança de um sistema de computador (PORRAS, 1992).

Uma desvantagem da maioria das abordagens baseadas em assinaturas é a inabilidade para descobrir ataques novos. A abordagem de STAT foi concebida para mitigar esta desvantagem, resumindo os detalhes dos ataques modelados. As ações são resumidas da forma nativa (por exemplo: registros de auditoria *standards* ou pacotes de cadeia) para uma representação de alto-nível, de forma que ações semelhantes em um sistema que possa ter representações de baixo nível diferentes seja traçado a um único tipo de ação.

Além disso, a metodologia de STAT apóia abordagens modeladas que representam só os passos (em uma intrusão) que são necessários para a efetividade do ataque. Resumindo, eliminando os detalhes de um ataque particular, é possível descobrir variações previamente desconhecidas de um ataque ou ataques que exploram mecanismos semelhantes.

No modelo de STAT original, foram representados ataques usando uma linguagem de diagrama de transição estatal informal, com estados representando instantes das propriedades das seguranças-pertinentes de um sistema e recursos, e transições representando as ações de assinatura críticas do ataque.

A implementação de USTAT inicial usou uma simples linguagem de regra que era difícil estender e até mesmo dirigir ataques de Solaris novos. Estender a linguagem de regra de USTAT para o domínio de cadeia, ou até mesmo para outros sistemas operacionais de *host*, era inviável, assim, a primeira implementação de NetSTAT foi representada com assinaturas de ataque em C, que foi compilado e unido ao IDS. Em ambos os casos, foram usados diagramas de transições estatais para desenvolver e argumentar sobre os ataques, mas estes diagramas foram traduzidos à mão à forma requerida pelo IDS designa-

do.

Como uma conseqüência, foi determinada que uma suíte de ferramenta baseado em STAT pudesse ser redesenhada como uma família de sistemas. O *toolset* compartilharia uma linguagem de domínio-independente comum, chamada STATL, que representa cenários de ataque, e um módulo núcleo para apoiar os mecanismos domínios-independentes, usando a *runtime* para emparelhar cenários de ataque contra um fluxo de eventos.

O STATL/core resultante desenha as características domínios-independentes das abordagens de STAT para capturas.

Os cenários de ataques no STAT são desenvolvidos por meio de um diagrama de estados que representa os estados e transições dos ataques. Esses canários são descritos a partir da linguagem de especificação STATL.

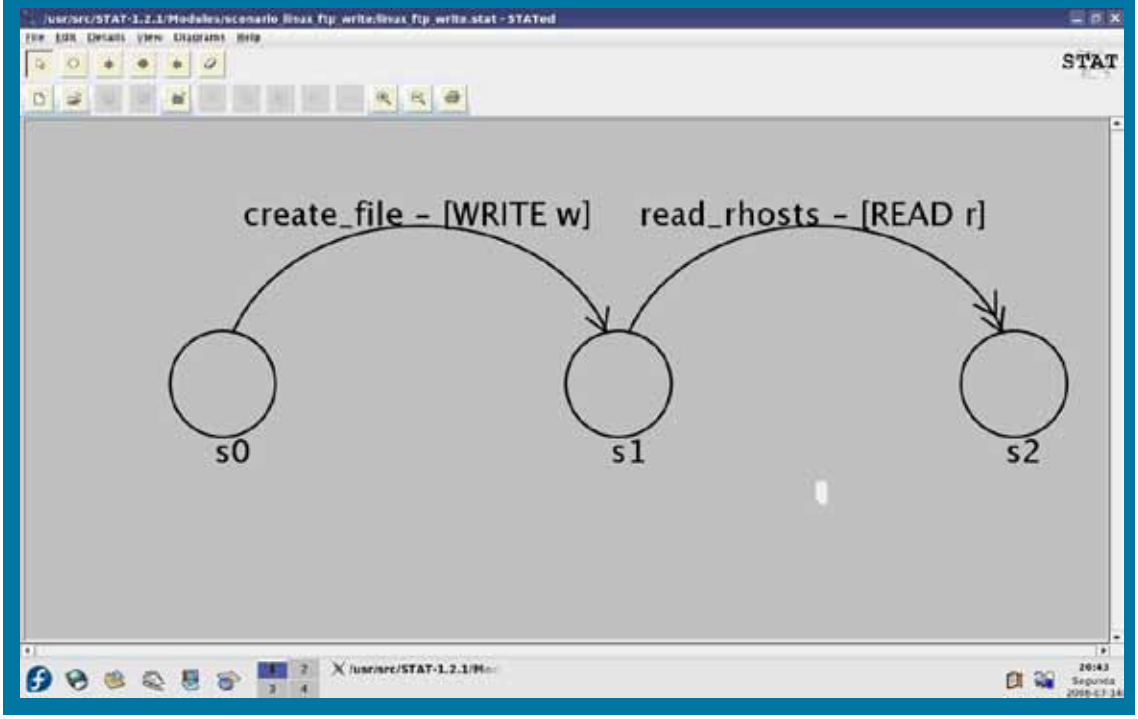
4.1 Cenários de ataques

Os cenários de ataques podem ser criados por meio do editor gráfico (Figura 1) onde os estados e as transições são representados por círculos e setas, respectivamente. São usados tipos diferentes de setas para denotar tipos diferentes de transições: um arco sólido com uma única ponta da flecha denota uma transição de *nonconsuming*, um arco sólido com uma ponta da flecha dupla denota uma transição *consuming* e um arco colidido denota uma transição *unwinding*. Esses tipos de transições serão detalhados posteriormente. Depois de criado o diagrama, o editor faz a tradução deste para uma descrição textual (Figura 2) em linguagem STATL do cenário descrito e o salva com a extensão stat.

O editor é utilizado para poder visualizar graficamente o cenário de ataque; pode-se simplesmente definir os cenários de ataques utilizando um editor de texto simples e escrever o cenário diretamente como mostrado na Figura 2.

Um cenário tem um nome, pode ter parâmetros, pode conter algum assunto de frente como constante e declarações de variáveis, e também contém os estados e transições que definem os passos do ataque. Variáveis em um cenário podem ser globais ou locais. Variáveis globais são compartilhadas por todas as instâncias do cenário (isto é, por todos os ataques daquele tipo). Variáveis locais são criadas em

Figura 1. Cenário representado por estados no editor



cada instância do cenário. Em outras palavras, quando uma variável global é atualizada, todas as instâncias do cenário vêem a atualização, mas quando uma variável local é atualizada, só a instância na qual a atualização acontece é afetada.

Os componentes principais de um cenário de STATL são estados e transições. Um estado tem um nome, uma afirmação, um bloco de código e uma anotação. O nome é um identificador único para o estado. Todos os outros elementos são opcionais. A afirmação estatal, se presente, é uma expressão lógica de C. Uma afirmação estatal é testada antes da entrada ao estado. O bloco de código estatal é executado depois de entrar no estado. Um bloco de código é um C que contém declarações. Por exemplo, na Figura 2, o bloco de código para estado s3 declara um username variável local e tem duas declarações e chama a extensão de USTAT procedimento userid2name e o tronco de stat de procedimento embutido. A afirmação e o código bloqueiam tenha acesso ao ambiente global (o jogo de variáveis globais), o ambiente local (o jogo de variáveis local àquela instância), e, no caso de um bloco de código, para as variáveis declaradas na declaração combinação. Uma anotação estatal é um conjunto de diretivas específicas a ser usadas no processo de tradução.

Uma transição tem um nome, um tipo, uma ação, uma afirmação, um bloco de código e uma anotação. O tipo da transição pode estar *consuming*, *nonconsuming*, ou *unwing*. Uma transição *nonconsuming* é usada para representar uma evolução do estado de um ataque acontecendo que não previne ocorrências adicionais de ataques que desovam do estado original. Então, quando uma transição de nonconsuming incendeia a fonte e estados de ataque de destino ficam válidos. Em contraste, o disparo de uma transição consuming faz a fonte declarar inválida a ocorrência de um ataque particular. Uma ação que apaga um arquivo, por exemplo, pode invalidar o estado de fonte. O arquivo não pode ser apagado duas vezes e então o estado original é nulo. Transições unwing representam uma forma de *rollback* e eles são usados para descrever eventos e condições que podem invalidar o progresso de um cenário de ataque e podem requerer o retorno a um estado mais cedo. Um evento de *logout*, por exemplo, pode fazer um cenário que envolve o usuário associado desinteressante e, então, o cenário pode ser trazido a um estado prévio, até mesmo antes do *login* de usuário.

A ação da transição é o elemento essencial de uma transição. Uma ação especifica um tipo de evento que pode causar o disparo da transição associada e um nome que se refere ao evento. Uma ação pode ser simples ou composta. Uma ação simples identifica um tipo de evento único e tem um nome único. Por exemplo, na Figura 2, a transição *create file* tem uma ação simples de tipo *write*, nomeada "w". Uma ação composta pode ser a conjunção ou a disjunção de ações. Além disso, ações podem ser aninhadas usando anotação de parêntese. A afirmação da transição, de bloco de código e anotação têm uma sintaxe e semântica que são semelhantes aos elementos estatais correspondentes. A diferença principal é que o espaço de nome inclui o nome de ação, e, então, a afirma-

Figura 2. Descrição textual de um cenário

```
use linux;
scenario linux_ftp_write
{
                    string CLASSIFICATION_NAME = "ftp_write";
                    string CLASSIFICATION_URL = "http://www.cs.ucsb.edu/~rsg/STAT";

                    // Actual User
                    string SOURCE_USERNAME = "unknown";
                    string SOURCE_USERID = "unknown";
                    string SOURCE_USERGROUP = "unknown";
```

(continuação)

```
                    // Process Keywords
                    string SOURCE_PROC_PATH = “unknown”;
                    string SOURCE_PROC_PID = “unknown”;

                    string ADDITIONAL_DATA_FILE = “unknown”;
                    string ADDITIONAL_DATA_DIR = “unknown”;


    int user;
    int pid;
    int inode;
    string objname;

    initial state s0 { }

    transition create_file (s0 -> s1) nonconsuming
    {
       [WRITE w] :
             (w.retcode >= 0) &&
             (w.owner != w.uid) &&
             !ignore_this(w.objname, w.act, stat)
       {
             inode = w.inode;
             objname = w.objname;
           }
    }

    state s1 { }

    transition read_rhosts (s1 -> s2) consuming
    {
       [READ r] :
             !strcmp(r.pathname, “/bin/login”) &&
             (r.retcode >= 0) &&
             (r.inode == inode)
           {
            ADDITIONAL_DATA_DIR = getDirname(r.objname);
            ADDITIONAL_DATA_FILE = getFilename(r.objname);
           }
    }

    state s2
    {
     {
       SOURCE_USERNAME = userid2name(user, stat);
                                    SOURCE_USERID = toString(user);
                                    SOURCE_PROC_PATH = objname;
                                    SOURCE_PROC_PID = toString(pid);
       log(“%d: user %s, program %s”, user, SOURCE_USERNAME.c_str(), objname.c_str());
     }
    }

}
```

ção da transição, bloco de código e anotação podem ter acesso aos campos da ação correspondente.

CONSIDERAÇÕES FINAIS

Ferramentas IDSs têm se mostrado extremamente importantes para uma arquitetura de rede segura. Como foi mencionado anteriormente, o presente trabalho teve como objetivo o estudo do IDS STAT e da linguagem de especificação de ataque, STATL.

Tratando-se de um software livre, surgiu a necessidade e a possibilidade de obter certo conhecimento intermediário em Linux.

Sendo as ferramentas IDSs um dos principais focos do estudo desse projeto, foi possível também adquirir considerável conhecimento acerca do funcionamento do software em questão, como métodos de detecção, tipos de arquiteturas, suas vantagens e desvantagens. Simultaneamente, foi possível, ainda, compreender os tipos de ataque e seus funcionamentos, oferecendo ênfase aos ataques DoS e DDoS.

Devido aos problemas com a configuração do IDS STAT, não foi possível a realização dos testes propostos no início da pesquisa, porém, verificou-se que as aplicações do Framework STAT, principalmente o STATED e a STATL, oferecem excelentes recursos para a otimização na geração das assinaturas de ataque.

Observou-se, finalmente, a possibilidade futura da realização de pesquisas mais aprofundadas e, provavelmente, bem sucedidas.

REFERÊNCIAS BIBLIOGRÁFICAS

AMBRÓSIO, D. R. (2002). **Métodos alternativos de reconhecimento de padrões para sistemas de detecção de intrusão**. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação - ICMC, Universidade de São Paulo - USP, São Carlos - São Paulo.

BALASUBRAMANIYAN, J. S.; FERNANDEZ, J. O. G.; ISACOFF, D.; SPAFFORD, E.; ZAMBONI, D. (1998). **An architecture for intrusion detection using autonomous agents**. Technical Report, Department of Computer Sciences, Purdue University, COAST Laboratory TR 98/05.

CERT.BR (2006). *Cartilha* **de segurança para Internet**. Disponível em <http://cartilha.cert.br/malware/>. Acesso em: abril de 2008.

HEADY, R.; LUGER, G.; MACCABE, A., SERVILA, M. (1990). **The Architecture of a Network Level Intrusion Detection System**, Technical Report - Department of Computer Science, University of New Mexico, USA.

KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a Internet: uma abordagem top-down**. 3. ed. São Paulo: Pearson-Addison Wesley, 2005.

LI, W.; ZHI-TANG L.; Yao L. (2006) **A novel algorithm SF for mining attack scenarios model**. Proceedings of the International Conference on e-Business Engineering. p. 55-61.

MOURA, J. A. B.; et al. **Redes de computadores: serviços, administração e segurança**. São Paulo: Makron Books, 1999.

NORTHCUTT, S.; ZELTSER, L.; WINTERS, S.; FREDERICK, K. K. ; RITCHEY, R. W. (2002). Desvendando segurança em redes usando Firewalls, VPNs, roteadores e sistemas de detecção de intrusão. Tradução de Daniel Vieira. Rio de Janeiro: Campus.

PAXSON, Vern. Bro: a system for detecting network intruders in realtime. Computer networks, vol. 31, p. 2435-2463, 1999.

PORRAS, P. STAT –A State Transition Analysis Tool for Intrusion Detection. Master's thesis, Computer Science Department, University of California, Santa Barbara, June 1992.

ROESCH, Martin. Snort – Lightweight intrusion detection for networks.

USENIX LISA Conference 1999. Seattle, WA.

VIGNA, G.; ECKMANN, S. and KEMMERER, R.. Designing and Implementing a Family of Intrusion Detection Systems. Reliable Software Group, Computer Science Department, University of California, Santa Barbara, 2003.

VIGNA, G.; ECKMANN, S. and KEMMERER, R.. STATL: AnAttack Language for State-based Intrusion Detection. Journal of Computer Security, 10(1/2):71{104, 2002.

XIANG, Y.; ZHOU, W.; LI Z. (2006). **An Analytical Model for DDoS Attacks and Defense**. Proceedings of the International Multi-Conference on Computing in the Global Information Technology. p.66.