UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

CHKMODEL: UM MECANISMO DE DEFESA CONTRA ATAQUES DDOS

Trabalho de Graduação

Aluno: Rafael Roque Aschoff (rra@cin.ufpe.br)
Orientador: Djamel Sadok (djamel@cin.ufpe.br)
Co-Orientador: Eduado Souto (esouto@gprt.ufpe.br)

Recife, 20 de janeiro de 2007

Resumo

Ataques de negação de serviço distribuídos (DDoS - *Distributed Denial-of-Service*) são uma verdadeira ameaça à Internet. Estes ataques são caracterizados pelo envio indiscriminado de pacotes e requisições a um determinado alvo, visando degradar a qualidade ou tornar complemente indisponíveis os serviços oferecidos pela vítima. A prevenção e o rastreamento dos ataques DDoS constituem operações de dificuldade elevada. Isso se deve ao grande número de máquinas atacantes envolvidas, ao uso de técnicas para forjar endereços IP e também à similaridade entre o tráfego legítimo e o tráfego de ataque. Essas dificuldades tornam a construção de ferramentas efetivas contra ataques DDoS uma tarefa desafiadora. Este trabalho propõe o projeto e a implementação de um novo esquema de detecção de ataques DDoS baseado na análise estatística do tráfego da rede. Este mecanismo foi denominado ChkModel (Check Model). Os resultados preliminares utilizando tráfego de rede real mostram que o mecanismo proposto é robusto e efetivo para os tipos de ataques avaliados.

Abstract

Distributed Denial-of-Service Attacks (DDoS) are a real threat to the Internet. These attacks are characterized by indiscriminate sending of packets and requisitions to a particular target aiming to degrade the quality or become complement unavailable the services offered by the victim. The identification, prevention and tracking of DDoS attacks represent operations of high difficulty due to the large number of attackers machines involved, the use of techniques to forge IP addresses, and the similarity between legitimate and attack traffic. These difficulties make the development of effective tools against DDoS attacks a task challenging. In this work, we propose a new mechanism for detecting DDoS attacks, named ChkModel, based on statistical analysis of network data. It can effectively detect anomalies with high detection rate, low false positives under the circumstance of using much fewer selected data. A series of initials experimental results on real dataset demonstrate the proposed mechanism is robust and effective against massive DDoS attacks.

Agradecimentos

Primeiramente, agradeço aos meus pais, Ronaldo Roque e Roseane Aschoff por me apoiarem, e se esforçarem ao máximo para me dar a educação necessária à minha formação pessoal.

Agradeço ao meu orientador, o professor Djamel Sadok, e à professora Judith Kelner por terem me dado uma oportunidade única de crescer como profissional ao trabalhar num ambiente tão enriquecedor como é o Grupo de Pesquisa em Redes e Telecomunicações (GPRT) do CIn-UFPE.

Agradeço à Universidade Federal de Pernambuco pelo apoio institucional e incentivo a minha formação acadêmica.

Por fim, um agradecimento especial ao meu co-orientador, Eduardo Souto, por contribuir de maneira decisiva na minha formação profissional e me ajudar nos momentos mais críticos da graduação.

Sumário

Índice de Figuras

1 Introdução

Ataques de negação de serviço distribuídos (DDoS – Distributed Denial-of-Service) são uma verdadeira ameaça à Internet. Estes ataques são caracterizados pelo envio indiscriminado de pacotes e requisições a um determinado alvo, visando degradar a qualidade ou tornar complemente indisponíveis os serviços oferecidos pela vítima.

Nos últimos anos, ataques DDoS têm obtido bastante importância, principalmente devido à sofisticação e coordenação na forma em que estes ataques são executados. Ferramentas como TFN, Trinoo e TFN2K têm sido empregadas pelos atacantes para sobrepujar os recursos do alvo de maneira relativamente fácil. A prevenção e o rastreamento dos ataques DDoS constituem operações de dificuldade elevada. Isso se deve ao grande número de máquinas atacantes envolvidas, ao uso de técnicas para forjar endereços IP (IP Spoofing), que escondem a origem verdadeira dos pacotes, e também à similaridade entre o tráfego legítimo e o tráfego de ataque. Essas dificuldades tornam a construção de ferramentas efetivas contra ataques DDoS uma tarefa desafiadora.

Como os ataques DDoS procuram tornar os recursos de um sistema indisponíveis para seus usuários, seja pela sobrecarga causada na vítima ou pela obstrução do canal de comunicação com os clientes da mesma, é essencial que estes ataques sejam detectados com um alto grau de precisão e o mais rápido possível, desta forma prejudicando o menor número de usuários legítimos possível. Isso pode ser feito se for possível detectar rapidamente alterações no uso dos recursos do sistema. Sendo assim, uma das formas de lidar com ataques dessa natureza é modelar o comportamento normal ou anômalo do uso de recursos do sistema através de métodos estatísticos. Por exemplo, Feinstein e Schnackenberg [1] propõem o uso de análises estatísticas de qui-quadrado para detectar essas anomalias. Manikopoulos [2] demonstra como um método estatístico chamado Aderson-Darling pode ser usado para detectar alterações no tráfego da rede. O método proposto em [3] usa distribuições ordenadas de freqüência de informações dos pacotes para calcular a entropia e usa desvios na entropia para indicar anomalias.

1.1 Objetivos

Este trabalho apresenta um Sistema de Detecção de Intrusão denominado ChkModel (Check Model), baseado na análise das estatísticas do tráfego da rede. O ChkModel foi projetado para comportar diferentes métodos de identificação e classificação de anomalias na rede. Este trabalho sugere o emprego de dois novos métodos que empregam técnicas comumente utilizadas para análise de processos estatísticos: EWMA(*Exponentially Weighted Moving Average*)[4] e CUSUM (Cumulative Sum)[5]. A idéia é que os métodos propostos possam cooperar para melhorar

a eficiência na detecção de ataques, aumentando a probabilidade de detecção e diminuindo a quantidade de alarmes falsos e tempo de detecção.

Para determinar a robustez e eficácia do mecanismo proposto, um conjunto de experimentos será realizado. A meta é fazer uso de um ambiente de rede real e controlado para gerar sinteticamente ataques DDoS com o objetivo de validar o ChkModel.

1.2 Estrutura do documento

O restante deste trabalho está dividido como segue. O Capítulo 2 descreve alguns conceitos básicos relacionados a Sistemas de Detecção de Intrusão e Ataques de Negação de Serviço. O Capítulo 3 apresenta alguns dos principais trabalhos que têm sido propostos na literatura com o objetivo de parar, evitar ou conter ataques na Internet. O Capítulo 4 descreve o ChkModel, seus componentes e métodos para identificação de anomalias. O capítulo 5 detalha os experimentos realizados e resultados obtidos. As conclusões e trabalhos futuros são apresentados no Capítulo 6. Por fim, são listadas as referências utilizadas na composição do documento.

2 Conceitos Básicos

Este capítulo descreve alguns conceitos básicos que foram necessários para o desenvolvimento teórico e aplicado no trabalho presente. Este capítulo está dividido em três partes: a primeira parte descreve em detalhes os sistemas de detecção de intrusão; a segunda referente aos principais conceitos de ataques de negação de serviços; por fim, a terceira parte apresenta de maneira resumida as principais técnicas de detecção de anomalias encontradas na literatura.

2.1 SDI - Sistemas de Detecção de Intrusão

O conceito de detecção de intrusão aparece pela primeira vez num artigo publicado por James Anderson[6]. Em seu trabalho, Anderson introduz a idéia de que trilhas de auditoria podiam conter informações importantes que podiam ser utilizadas no rastreamento de usos impróprios do sistema e ainda no entendimento do comportamento do usuário.

Um SDI, ou Sistema de Detecção de Intrusão (NIDS – Network Intrusion Detection System) é uma ferramenta de software que tem por finalidade detectar acessos não autorizados a um computador ou uma rede [7]. Estes sistemas são capazes de detectar uma grande variedade de comportamentos maliciosos. Isto inclui ataques contra serviços vulneráveis, acesso não autorizado a determinados arquivos, *logins* não autorizados e códigos maliciosos (*malware*). De modo geral esses sistemas podem ser entendidos como entidades de monitoramento dinâmico que complementam as habilidades de monitoramento estático de um *firewall*.

Conceitualmente, existem dois diferentes tipos de SDI, os baseados em rede e os baseados em *host*, cada um com suas particularidades. Sistemas de *host* monitoram dados em uma determinada máquina, sejam eles processos, sistemas de arquivos ou uso de CPU. SDIs baseados em Rede (SDIRs) monitoram todo o tráfego trocados entre *hosts*, ou seja, todos os dados que passam através da rede.

2.1.1 Modelo Conceitual

Podemos identificar alguns elementos comuns nos projetos de SDIs, independente do seu tipo ou métodos aplicados. Esses elementos representam atividades essenciais desempenhadas na identificação das ações consideradas maliciosas. Através de um esforço no sentido de padronizar a nomenclatura e a função de cada um desses componentes em [9] é proposto um modelo chamado CIDF (Common Intrusion Detection Framework) ilustrado na Figura 2-1.

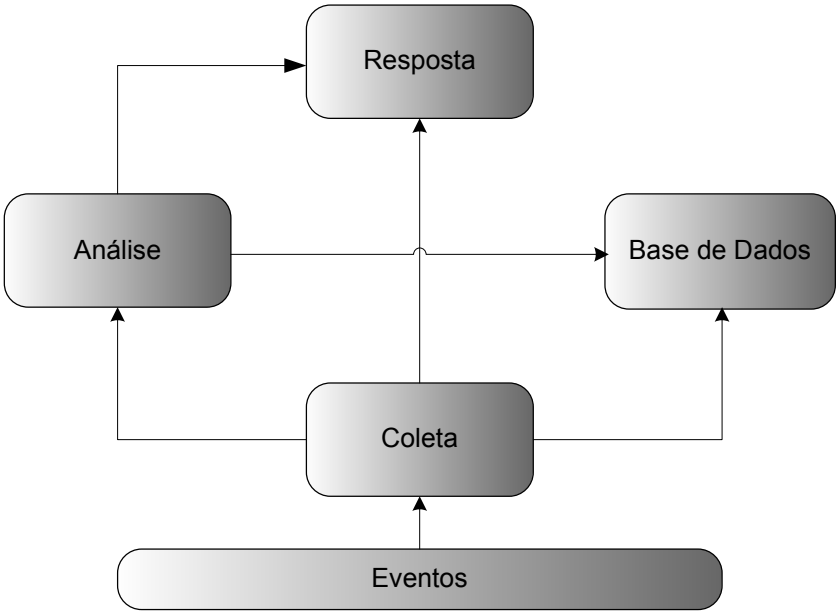

Figura 2-1: Modelo conceitual de uma ferramenta de IDS.

Capturador de Eventos - Este componente é utilizado na fase de coleta de dados. Os eventos observados são padronizados e enviados para um componente de processamento. As fontes de dados podem ser variadas, incluindo *logs* das atividades de rede, registro de comando do sistema, entre outros.

Analisador de Eventos – O analisador de eventos recebe dados provenientes dos geradores de eventos e é responsável pela detecção da intrusão. Por este motivo esta etapa é considerada o coração de um sistema de detecção de intrusão. É aqui onde os algoritmos de detecção são implementados. Normalmente estes algoritmos são classificados em três grandes grupos: baseados em assinatura, anomalia ou híbridos.

Base de Dados de Eventos - É comum que os dados coletados sejam armazenados por um longo período de tempo para futura análise. O volume de dados normalmente é muito grande, o que se traduz em assunto de pesquisa e questão a ser levada em forte consideração no projeto de sistemas de detecção de intrusão.

Unidade de Resposta - Todo processo de saída do sistema e de responsabilidade da Unidade de Resposta. A saída pode ser tanto uma resposta automática direta a uma intrusão, como terminar uma conexão ou reiniciar um processo, como pode ser o envio de um alerta ao administrador do sistema.

2.1.2 Alarme

Uma intrusão pode ser definida como qualquer conjunto de ações que tentam comprometer a integridade, confidencialidade ou disponibilidade dos dados ou sistemas [10].

Anderson [6] apresenta uma modelo que classifica as intrusões como: penetrações externas, penetrações internas e transgressões. Penetrações externas são definidas como intrusões que são realizadas por usuários não autorizados a utilizar o sistema; penetrações internas são aquelas que são efetuadas pelos usuários autorizados que não estão autorizados a acessar os dados em questão; transgressão é definida como uso não autorizado tanto do sistema como de seus dados. Na prática, as seguintes situações, sugeridas em [11] podem ocorrer:

- **Atividade intrusiva, porém normal**: esta situação é extremamente grave, pois gera a situação de falso-negativo, ou seja, o sistema não detecta um ataque ou intrusão.
- **Atividade não intrusiva e normal**: situação denominada positivo-negativo. Atividade normal e não intrusiva, portanto, não será detectada pelo sistema.
- **Atividade não intrusiva e anômala**: situação denominada falso-positivo. O sistema alerta indicando uma intrusão que na verdade não ocorreu.
- **Atividade intrusiva e anômala**: situação ideal em que o sistema detecta corretamente o ataque ou intrusão.

2.1.3 Classificação do SDI

Historicamente, sistemas de detecção de intrusão têm concentrado esforços na fase de análise. Os sistemas de detecção de intrusão podem ser classificados quanto à tecnologia empregada nessa fase.

Análise de Assinatura – Sistemas de Detecção de Intrusos baseados na analise de assinatura ou por uso indevido ("*misuse detection*") são sistemas que procuram por padrões de ataques e intrusões previamente conhecidos. São também conhecidos como sistemas de detecção de intrusão baseado em conhecimento. A premissa básica dos SDIs que se enquadram nesta classe é que existem ataques com características precisas, bem definidas e que podem ser facilmente codificadas em um sistema especialista. Esses sistemas possuem uma base de assinaturas de ataques conhecidos que deve ser constantemente atualizada a medida que novos ataques sejam descobertos. Possíveis exemplos de assinaturas são:

- Endereço IP de origem participar da faixa de endereços reservados 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16. Neste caso só e preciso verificar o endereço de origem do cabeçalho IP.
- Segmentos TCP com o *flag* SYN ativado para determinadas portas, por exemplo, portas 21, 22 e 23. Uma simples verificação nos *flags* do cabeçalho TCP poderia analisar a regra.
- Tentativa de acesso do usuário *root* a um servidor FTP, no qual não é permitida sua entrada. É necessário verificar o cabeçalho dos pacotes TCP certificando-se de que é um tráfego destinado à porta 21 do servidor. Ainda precisa-se analisar se o pacote carrega a informação "USER root".

Os três exemplos apresentados ilustram regras de um SDI baseado em redes. Além disso as técnicas de análise de assinaturas também podem ser baseadas em eventos relacionados a um computador ou sistema específico (baseado em *host*). Um exemplo de regra, neste caso, seria gerar um alerta se o percentual de ocupação do processador superasse determinado valor por um intervalo de tempo específico, uma determinada seqüência de chamadas a funções do sistema operacional ocorresse ou determinados arquivos críticos fossem alterados.

É importante ressaltar que, independentemente do foco (rede ou *host*) os sistemas baseados em analise de assinaturas necessitam de conhecimento especialista humano para operar. Um dos principais benefícios dessa técnica é que ataques conhecidos podem ser detectados com uma taxa muito baixa de falsos positivos. Outro benefício é que a técnica permite que o sistema fique protegido imediatamente após sua instalação. A principal desvantagem dos sistemas de detecção baseados em assinaturas é a sua incapacidade de detectar novos ataques, ou mesmo, ataques que ainda não façam parte da sua base de assinaturas e padrões. Este é o principal fator motivador para pesquisa e desenvolvimento de sistemas de detecção de intrusos baseado em anomalias.

Análise de Anomalia – Sistemas de detecção de intrusão baseados na análise de anomalia, também chamados sistemas de detecção de intrusão baseados em comportamento, procuram determinar ou criar modelos que representem o comportamento normal ou esperado do sistema computacional ou rede em análise e alertam sempre que desvios neste comportamento esperado forem encontrados.

Sistemas dessa categoria oferecem vários benefícios. Em primeiro lugar, eles possuem a capacidade de detectar ataques internos. Por exemplo, se um usuário ou alguém usando uma

conta roubada inicia a execução de ações que estão fora do padrão normal desse usuário, um sistema de detecção de anomalia pode gerar um alarme. Em segundo lugar, devido ao sistema ser baseado em perfis personalizados, é difícil para um atacante saber com certeza o que ele pode realizar sem disparar um alarme. Em terceiro lugar, um sistema de detecção de anomalia tem a capacidade de detectar ataques previamente desconhecidos. Isto se deve ao fato de que um perfil de atividade intrusiva não é baseado em assinaturas específicas. Uma atividade intrusiva gera um alarme porque se afasta da atividade normal, não porque alguém tenha configurado o sistema para procurar por uma assinatura de ataque específica.

Por outro lado, estes sistemas também sofrem de várias desvantagens. A primeira desvantagem evidente é que deve passar por um período de treinamento no qual os perfis de usuários normais são criados através da definição de perfis de tráfego "normal". Além disso, criar um perfil de tráfego normal é uma tarefa difícil. A criação desse perfil de maneira inadequada pode levar a um desempenho indesejado. A manutenção desses perfis também pode ser uma tarefa que consuma muito tempo. Uma vez que sistemas de detecção de anomalia estão procurando por acontecimentos anômalos ao invés de ataques, estes estão sujeitos aos alarmes falsos. Por fim, uma armadilha destes sistemas acontece quando um usuário malicioso consegue gradualmente treinar o sistema para entender o comportamento malicioso como normal.

Análise Híbrida - Sistemas híbridos ou compostos combinam as duas abordagens apresentadas. Um sistema híbrido atua inspirado num sistema de detecção de intrusão por assinatura que toma suas decisões através de um modelo híbrido, ou seja, leva em consideração tanto o comportamento normal do sistema como o comportamento intrusivo dos atacantes.

2.1.4 Técnicas da Análise de Anomalia

Uma técnica de detecção de anomalia usualmente consiste de duas fases: a fase de treinamento e a fase de teste. Na primeira, o perfil ou modelo de tráfego normal é criado. Na segunda, o modelo é aplicado aos novos dados coletados. A premissa básica é que atividades de intrusão fazem parte do subconjunto composto por atividades anômalas [11]. Idealmente, o conjunto de atividades maliciosas será igual ao conjunto de atividades anômalas. Nesta situação o sistema não gerará nenhum falso-positivo e nenhum falso-negativo.

De forma geral, as técnicas de detecção de anomalia podem ser categorizadas em três grandes grupos: baseados em estatística, aprendizagem de máquina e mineração de dados, os quais são apresentados a seguir.

Métodos Estatísticos - Em métodos estatísticos para detecção anomalia, o sistema observa a atividade dos indivíduos e gera perfis para representar o seu comportamento. O perfil tipicamente inclui medidas como intervalo de tempo médio de chegada de pacotes, vazão esperada da rede, consumo de memória, entre outros. Tipicamente, dois perfis são mantidos para cada elemento analisado: o perfil atual e o armazenado. Durante o processamento do sistema, o método empregado calcula o desvio do perfil atual comparando-o com o esperado, um desvio elevado no comportamento normal pode ser identificado como suspeito.

As abordagens estatísticas para detecção de anomalia têm uma série de vantagens. Em primeiro lugar, como a maioria dos sistemas de detecção de anomalia não exige conhecimento prévio de falhas de segurança ou de ataques, a solução e eficiente em detectar ataques muito recentes. Além disso, abordagens estatísticas podem fornecer notificações exatas das atividades maliciosas que ocorrem normalmente durante longos períodos de tempo e são bons indicadores de ataques de negação de serviço iminentes. Um exemplo muito comum desse tipo de atividade é um *portscan*. Normalmente, a distribuição de *portscans* é altamente anômala em comparação com o tráfego normal. Com isto em mente, mesmo *portscans* que são distribuídos durante um longo período de tempo serão registrado visto que são inerentemente anômalos. No entanto, esquemas de detecção de anomalia baseados em estatística também têm seus inconvenientes. Pode ser difícil determinar limiares para equilibrar o risco de falsos positivos com o risco de falsos negativos. Além disso, os métodos estatísticos necessitam de distribuições estatísticas precisas, porém, nem todos os comportamentos podem ser modelados usando métodos puramente estatísticos.

Máquina de Aprendizagem - Máquina de aprendizagem pode ser definida como a capacidade de um programa ou sistema em aprender e melhorar seu desempenho sob certas tarefas ou grupo de tarefas ao longo do tempo. Os IDS baseados em máquinas de aprendizagem visam responder as mesmas questões resolvidas pelas abordagens que usam técnicas estatísticas ou de mineração de dados. Entretanto, diferente das abordagens estatísticas que tendem a focar sobre o entendimento do processo que gera os dados, as técnicas de máquina de aprendizagem focam sob a construção de um sistema que melhora seu desempenho em função dos resultados anteriores. Em outras palavras, esses sistemas são capazes de mudar sua estratégia de execução com base na informação adquirida recentemente.

2.2 Ataques de Negação de Serviços

De acordo com a definição do CERT (**C**omputer **E**mergency **R**esponse **T**eam)[12], os ataques DoS (**D**enial **o**f **S**ervice), também denominados Ataques de Negação de Serviços, consistem em tentativas de impedir usuários legítimos de utilizarem um determinado serviço de um computador. Para isso, são usadas técnicas que podem: sobrecarregar uma rede a tal ponto em que os verdadeiros usuários dela não consigam usá-la; derrubar uma conexão entre dois ou mais computadores; fazer tantas requisições a um site até que este não consiga mais ser acessado; negar acesso a um sistema ou a determinados usuários.

Explicando de maneira ilustrativa, imagine você usa um ônibus regularmente para ir ao trabalho. No entanto, em um determinado dia, uma quantidade enorme de pessoas "furaram a fila" e entraram no veículo, deixando-o tão cheio que você e os outros passageiros regulares não conseguiram entrar. Ou então, imagine que você tenha conseguido entrar no ônibus, mas este ficou tão cheio que não conseguiu sair do lugar por excesso de peso. Este ônibus acabou negando o seu serviço - o de transportá-lo até um local -, pois recebeu mais solicitações - neste caso, passageiros - do que suporta. Os ataques de negação de serviços atuam de maneira semelhante visando impedir usuários legítimos de fazer uso de um serviço. Portanto, é importante frisar que quando um computador ou site sofre um ataque DoS, ele não é invadido, mas sim sobrecarregado. Isso independe do sistema operacional utilizado.

Os ataques do tipo DoS mais comuns podem ser feitos devido a algumas características do protocolo TCP/IP (***T****ransmission* ***C****ontrol* ***P****rotocol* / ***I****nternet* ***P****rotocol*) [8], sendo possível ocorrer em qualquer computador que o utilize. Uma das formas de ataque mais conhecidas é a *SYN Flooding*, onde um computador tenta estabelecer uma conexão com um servidor através de um sinal do TCP conhecido por SYN (***Syn****chronize*). Se o servidor atender ao pedido de conexão, enviará ao computador solicitante um sinal chamado ACK (***Ack****nowledgement*). O problema é que em ataques desse tipo, o servidor não consegue responder a todas as solicitações e então passa a recusar novos pedidos.

Outra forma de ataque comum é o UDP Packet Storm, onde 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 suas funções.

A RFC 4732 (Internet Denial-of-Service Considerations) [13] afirma que devido a falta de consideração a ataques de negação de serviço na concepção da arquitetura da Internet, praticamente todos os serviços da rede mundial de computadores são vulneráveis a ataques deste tipo quando em suficiente escala. Na maioria dos casos, esta escala pode ser obtida através do

comprometimento da vários *hosts* ou roteadores que trabalhando em conjunto passam a efetuar um ataque de negação de serviço distribuído (DDoS – *Distributed Denial of Service*).

O primeiro registro de atividades DDoS deve-se ao CERT, em 22 de Julho de 1999 [14]. Entretanto, os ataques DDoS ganharam a atenção do público apenas no primeiro quadrimestre de 2000, quando um grande número de sites como o Yahoo, EBay, Amazon e CNN ficaram inoperantes devido a ataques dessa natureza [15]. No Brasil, no mesmo período, teve-se notícia de ataques contra sites como: UOL, Globo On e IG. A primeira versão do *worm* Code Red, desenvolvida em 2001 tinha intenção de elaborar um ataque DDoS com todas as máquinas infectadas à Casa Branca, nos Estado Unidos [16]. Ataques deste tipo fizeram história pelo número de computadores envolvidos. O *worm* MyDoom ficou famoso ao lançar em 2004 um enorme ataque contra o SCO Group [17]. O ataque teve um impacto tão grande que a SCO tomou a decisão de seu derrubar o servidor e mantê-lo fora do ar até que o *worm* desativasse sua.

2.2.1 O Ataque DDoS

A Figura 1 retrata uma arquitetura genérica para ataques de negação de serviço distribuídos. Pode-se perceber que este tipo de ataque é composto por diferentes elementos. Nesta monografia será utilizada a seguinte nomenclatura:

- **Atacante**: Aquele que coordena o ataque.
- **Mestre**: Máquina que recebe os parâmetros para o ataque e comanda os Zumbis (veja a seguir).
- **Zumbi**: Máquina que efetivamente concretiza o ataque contra a(s) vítima(s).
- **Vítima**: Alvo do ataque.
- **Cliente**: Aplicação das máquinas Mestre que controla os ataques através de comandos enviados às máquinas Zumbi.
- **Daemon**: Processo que roda nas máquinas Zumbi, responsável por receber e executar os comandos enviados pelo Cliente.

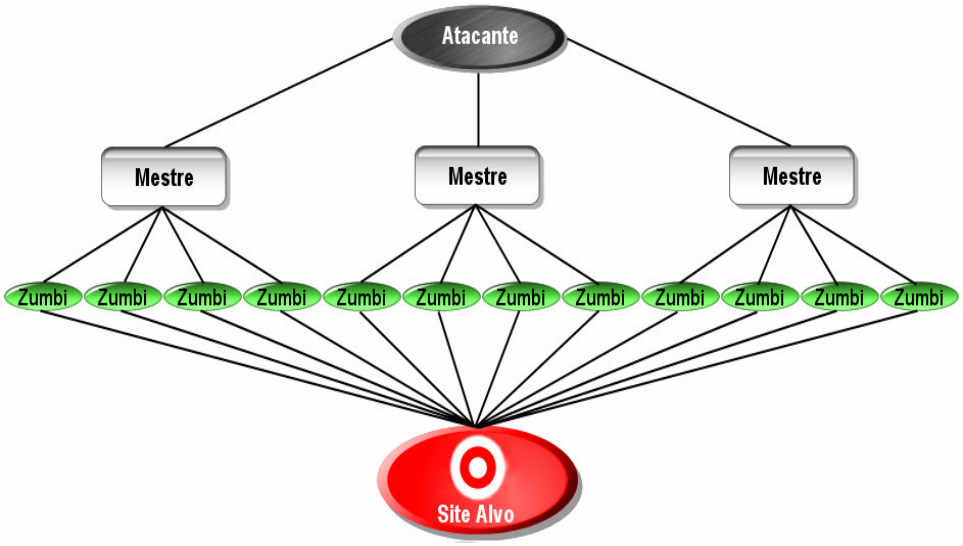

Figura 2-2: Estrutura de um ataque de negação de serviço distribuído.

Ataques DDoS acontecem em três fases principais: a fase de intrusão em massa, onde ferramentas são utilizadas a fim de comprometer e tomar o controle de máquinas remotas; uma fase onde o atacante instala os programas necessários nas máquinas invadidas; e por fim, a fase onde é consolidando efetivamente o ataque. A seguir são apresentados detalhes de cada uma destas fases:

Fase 1 - Intrusão em Massa: Na primeira fase é feito um recrutamento das máquinas que irão participar do ataque. Este processo normalmente é realizado de forma automática através da busca por vulnerabilidades em máquinas remotas que permitam seu controle. Estas brechas de segurança são então exploradas e o processo gera uma lista com o endereço das máquinas que foram afetadas e que serão utilizadas no ataque.

Fase 2 - Instalação das Ferramentas DDoS: Nesta segunda fase é feita a instalação das ferramentas DDOS e escolha dos mestres e zumbis com o intuito de montar a rede de ataque. Depois que a fase de intrusão em massa é concluída as máquinas recebem os programas necessários para efetuar o ataque, onde estão incluídos os programas de coordenação. Após as ferramentas necessárias estarem instaladas, as máquinas são divididas entre mestres e zumbis pelo atacante. Normalmente são escolhidos para ser mestres aquelas máquinas com menor controle de atividades suspeitas, já os zumbis são máquinas mais velozes e de banda larga. Neste momento, os mestres passam a rodar a aplicação cliente e as os zumbis o *daemon*. É bastante comum que

sejam instaladas ferramentas para impedir ou dificultar a detecção dos programas instalados. Estas ferramentas são comumente conhecidas como rootkits.

Fase 3 - Ataque: Como observado na Figura 2-1, em um ataque DDoS, o atacante tem sobre seu controle várias máquinas mestre que trabalham diretamente pra ele. Essas máquinas, por sua vez, coordenam os zumbis subordinados a elas. Assim que o atacante ordena o ataque a uma ou mais vítimas, os mestres, através do programa cliente, comunicam-se com os zumbis, através do daemon, que efetivamente operam o ataque, inundando a vítima com requisições inválidas.

2.2.2 Ferramentas usadas em Ataques DDoS

Ataques DDoS atingiram um grau de sofisticação que é possível a qualquer pessoa efetuar um ataque complexo e em larga escala através da utilização de várias ferramentas DDoS, facilmente encontradas em sites e fóruns *hacker*. A primeira ferramenta que se tem notícia, desenvolvida para efetuar ataques dessa natureza surgiu em 1998. A partir daí muitas ferramentas foram desenvolvidas, cada vez com mais sofisticação e facilidade de uso. Abaixo são apresentadas algumas das mais conhecidas ferramentas utilizadas em ataques DdoS.

- **Trinoo** [18]: O Trinoo é um projeto de DoS, com um conjunto de programas do tipo mestre/escravo que implementam uma ferramenta de DDoS. É usada para lançar ataques coordenados, especificamente, ataques do tipo UDP flood. Uma rede Trinoo é composta por um número pequeno de mestres e um grande número de zumbis. O atacante se comunica com os mestres via TCP enquanto estes comunicam-se com os zumbis via UDP.
- **TFN** [18]: Triple Floof Network é uma ferramenta que usa uma arquitetura diferente da apresentada pelo Trinoo. A comunicação entre os diferentes componentes do ataque não utiliza o protocolo TCP ou UDP, ao invés disso são utilizados apenas pacotes ICMP. Em adição ao ataque realizado pelo Trinoo, com o TFN pode-se também efetuar ataques do tipo SYN flood, ICMP flood e Smurf/Fraggle.
- **Stacheldraht** [19]: Combina características do Trinoo e do TFN e surgiu em setembro de 1999. Assim como o TFN, o Stacheldraht consegue efetuar ataques do tipo UDP flood, SYN flood, ICMP flood e Smurf/Fraggle e utiliza ICMP para comunicação entre mestres e zumbis, contudo, a comunicação entre o atacante e os mestres é feita pelo protocolo TCP de forma criptografada.

- **Shaft** [21]: O Shaft combina características do Trinoo, TFN e Stacheldraht. Tem recursos de coleta de estatísticas e uma característica bem interessante: todos seus pacotes TCP têm número de seqüência fixo.
- **TFN2K** [22]: A ferramenta Tribe Flood Network 2000, ou ainda, Tribe Flood Network 2k é uma versão aprimorada do TFN, sendo as duas versões desenvolvidas por Mixter. Além do daemon ser instruído para alternar entre os tipos de ataques suportados pelo TFN, esta versão aprimorada adiciona características para dificultar sua detecção e melhorar o controle remoto da rede de zumbis.
- **Trinity** [23]: Primeira ferramenta DDoS controlada via IRC. Cada zumbi, depois de infectado pelo Trinity, conecta-se a um canal e espera comandos para realizar o ataque.

3 Trabalhos Relacionados

Nos últimos anos, muitas pesquisas têm sido dedicadas ao problema de parar, eliminar ou mitigar ataques na Internet. Este capítulo não pretende apresentar todas as abordagens existentes, visto que a lista de soluções é extensa e continua crescendo, mas apresenta algumas das principais abordagens para detecção de intrusão disponíveis na literatura.

3.1 IP TRACEBACK

Na Internet, o identificador de origem de um pacote é o endereço IP, mas pode ser difícil determinar a origem verdadeira dos pacotes devido a técnicas que falsificam o endereço de origem desses pacotes (*IP Spoofing*). *IP Traceback* é o nome genérico dado aos métodos que procuram determinar, de maneira confiável, a origem de um pacote. A seguir é apresentada uma solução encontrada na literatura para o problema de identificação da fonte.

3.1.1 ICMP Traceback

Bellovin [24] propõe um novo esquema de rastreamento baseado no uso explícito de mensagens ICMP gerada pelos roteadores. A idéia nesse esquema é simples, para cada roteador é encaminhada uma mensagem ICMP especial incluindo informações a respeito dos roteadores adjacentes ao longo de um caminho para um destino. Durante um ataque do tipo flooding, o host vítima pode então fazer uso destas mensagens para reconstruir o caminho de volta até o atacante. Este esquema trás alguns benefícios se comparado com as outras abordagens. Entretanto, existem várias desvantagens em seu projeto atual que complicam sua real implementação. A principal delas é que os roteadores comumente bloqueiam mensagens ICMP devido a questões de segurança associadas com este tipo de tráfego. Além disso, o atacante poderia forjar uma mensagem ICMP, conduzindo a um erro de rastreamento.

3.2 Pushback

Pushback [24] é um mecanismo de defesa contra ataques de negação de serviços distribuídos que trata os ataques como um problema de controle de congestionamento.

Basicamente, existem duas fases para este mecanismo [RIP - 12]: Controle de congestionamento agregado (CCA) e *pushback*. O CCA detecta o congestionamento no roteador através de uma assinatura de ataque que é traduzida para um filtro no roteador. Cabe ao algoritmo

Pushback propagar para os roteadores vizinhos a taxa de dados que um *host* poderá operar e desta forma eliminar os pacotes que provavelmente pertencem a um ataque.

Este mecanismo apresenta um bom desempenho contra ataques DDoS do tipo *flooding* (inundação). Contudo, ao basear suas decisões a partir do ponto de vista do controle de congestionamento, uma especificação menos restrita das assinaturas ou políticas pode levar o tráfego legítimo a ser limitado, enquanto uma especificação mais restritiva pode permitir que os atacantes contornem a proteção.

3.3 D-WARD

O D-WARD [25] foi desenvolvido como um sistema regulador de tráfego para prevenir ataques DDoS a partir de sua fonte. O D-WARD detecta ataques através da monitoração constante dos fluxos de saída da rede, comparando-os com modelos pré-estabelecidos de fluxos baseado no comportamento normal dos protocolos de transporte e aplicação, identificando assim tráfegos legítimos, suspeitos e de ataque. Um regulador de tráfego é utilizado para aplicar limitações de banda ao fluxos considerados como suspeitos ou ilegítimos. Conforme mostrado na Figura 3-1, o D_WARD é composto pelos componentes de observação (observation component) e modelador de tráfego (throttling component).

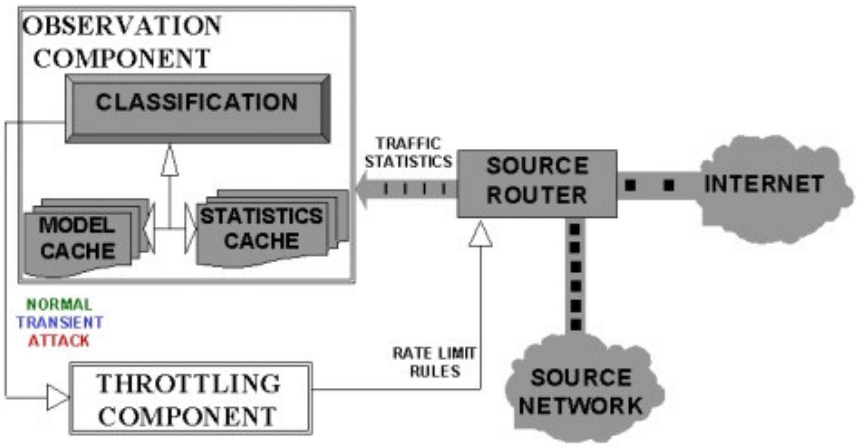

Figura 3-1: Arquitetura do Sistema D-WARD.

O sistema D-WARD deve ser configurado com uma lista de endereços cuja saída do tráfego deve ser monitorada. Cabe ao componente de observação monitorar todos os pacotes e coletar estatísticas sobre o tráfego originado ou destinado aos hosts indicados na lista. Periodicamente,

essas estatísticas são comparadas ao modelo de tráfego normal e os resultados são encaminhados para o componente responsável por regular o tráfego.

O D-WARD foi projetado para atuar próxima a fonte dos pacotes. Os autores argumentam que desta forma os recursos da Internet (largura de banda, recursos dos roteadores, etc.) são preservados. Um ponto importante é que a precisão de detecção deste mecanismo depende dos modelos de tráfego normal pré-definidos.

3.4 NetBouncer

O NetBouncer [26] é um abordagem que procura legitimar os clientes e descartar o trafego não legitimo. A identificação e feita através de vários testes de legitimidade, que geram uma lista de clientes legítimos e autenticados. Uma nesse grupo o cliente passa a ter seu trafego privilegiado em detrimento de clientes ainda não autenticados. A lista expira em intervalos de tempo constante com o objetivo de evitar que um atacante forje uma autenticação e a use para atacar.

Apesar de conseguir de forma eficiente priorizar os clientes autenticados o NetBouncer assume em seus testes algumas características que não estão presentes em todos os clientes legítimos, por exemplo, a capacidade de responder a comandos ping. Comprovadamente nem todos usuários podem responder a esse tipo de comando, em especial aqueles que estão localizados atrás de um firewall configurado para bloquear esse tipo de trafego.

3.5 SOS – Security Overlay Services

A arquitetura SOS (Security Overlay Services) [28] é um tipo de rede sobreposta (overlay network) composta por nós que se comunicam entre si usando a rede subjacente (rede IP). A figura x mostra uma visão geral da arquitetura SOS. Fundamentalmente, o objetivo da infra-estrutura SOS é distinguir o tráfego autorizado e não-autorizado, permitindo que apenas o primeiro alcance o destino, enquanto que o ultimo deve ser descartado (ou ter sua taxa reduzida). Dessa forma, a rede SOS se torna um grande firewall distribuído que distingue o tráfego malicioso e legítimo.

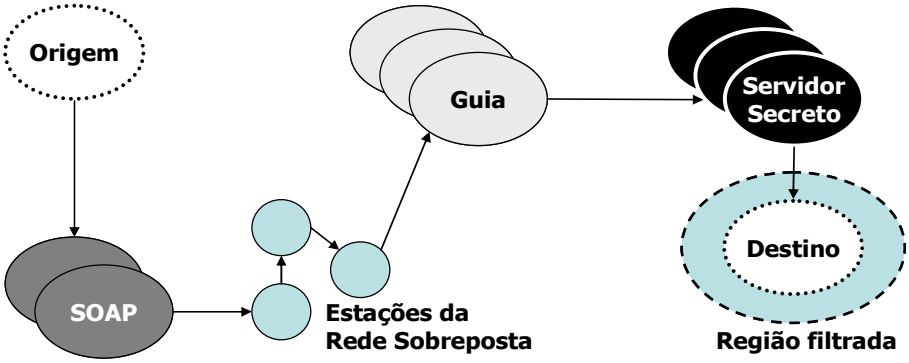

Figura 3-2: A arquitetura básica do SOS.

De maneira geral, a arquitetura foi projetada para trabalhar em redes que estão sob um firewall, que fica responsável por filtrar os pacotes que não pertencem as conexões confiáveis. Uma desvantagem dessa abordagem e que o firewall pode ficar inoperante devido a ataques em massa, prejudicando a eficiência da solução.

4 ChkModel

O ChkModel foi projetado para ser um Sistema de Detecção de Intrusão baseado em rede. O objetivo principal do sistema é monitorar continuamente o tráfego de um segmento de rede em busca de ataques. A Figura 4-1 apresenta a arquitetura do sistema proposto. Como se pode perceber a partir de uma análise desta figura, a arquitetura do ChkModel é bem semelhante ao modelo CIDF [9] apresentado no Capítulo 2.

As funcionalidades básicas de cada um dos componentes seguem o modelo CIDF e seus detalhes são apresentados a seguir.

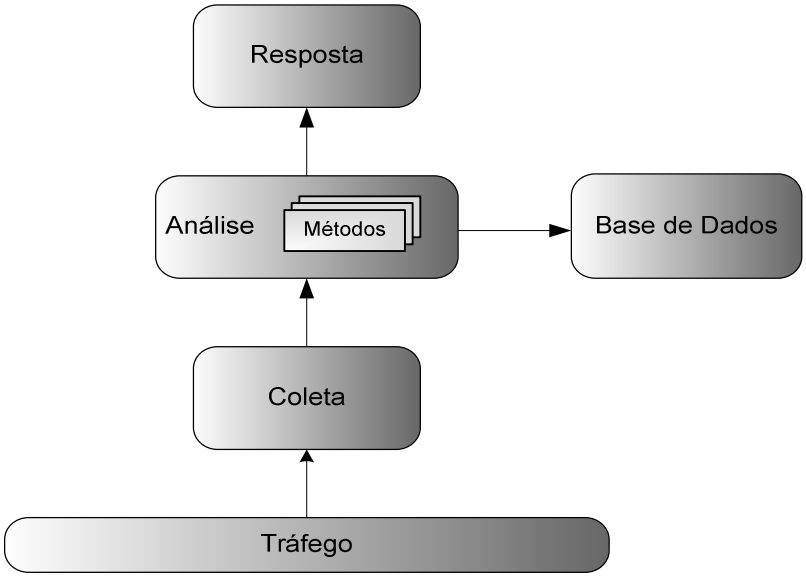

Figura 4-1: Arquitetura do ChkModel.

4.1 Descrição dos Componentes

Como mostra a Figura 4-1, o ChkModel é composto pelos seguintes componentes: Coleta, Análise, Base de Dados e Resposta. O componente de coleta é responsável por capturar os pacotes, estruturá-los e encaminhá-los ao componente de análise, cuja função é encaminhar os pacotes estruturados para uma lista de métodos registrados que são responsáveis pela classificação do tráfego. A base de dados serve para guardar as informações necessárias a cada um dos métodos de classificação. Por fim, o componente de resposta é responsável por receber as sinalizações feitas pelo componente de análise e realizar uma ação. Os detalhes de cada um dos componentes são descritos a seguir.

4.1.1 Coleta

O componente de coleta monitora todo o tráfego de entrada e saída do roteador em que está instalado. O monitoramento do tráfego pode ser realizado tanto através de captura em tempo real dos pacotes da interface de rede em questão como também através da leitura de traces gerados por ferramentas de captura de pacotes conhecidas como o Tcpdump [29]. A biblioteca JPCAP [30] foi escolhida para programar as funções relacionadas à captura de pacotes, bem como estruturação dos mesmos. Quando os pacotes chegam no ponto de coleta, estes são identificados com base no protocolo de comunicação utilizado. Os *bytes* são então convertidos para o seguintes formatos suportados: Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, e ICMPv4. Este procedimento facilita a análise realizada no componente de análise, visto que as informações podem ser acessadas de maneira padronizada e intuitiva.

4.1.2 Análise

O Componente de análise é formado por um repositório de métodos que podem ser utilizados para detecção de ataques. A utilização de múltiplos componentes de classificação é uma estratégia que procura contornar a dificuldade de se modelar o comportamento da Internet com apenas uma variável controlada ou um processo estatístico [7]. Este trabalho utiliza dois métodos de classificação baseado em técnicas estatísticas TCPModel e SIPModel. Detalhes desses métodos são apresentados a seguir.

4.1.2.1 TCPModel

Por definição, um protocolo especifica o formato dos dados e os procedimentos (mensagens) que dois computadores trocam para possibilitar a transferência de dados. No caso do TCP (*Transmission Control Protocol*)[8], ele apresenta como principais características a transferência de dados confiável fim-a-fim (os bytes transmitidos precisam ser reconhecidos, há recuperação de dados perdidos, descarte de dados duplicados e reorganização dos dados recebidos fora de ordem) e a comunicação bidirecional (*full-duplex*) entre cliente e o servidor. Baseado neste conceito é possível dizer que um comportamento de taxa de envio agressiva de pacotes, combinado com uma baixa taxa de resposta, corresponde a uma comunicação anômala [26].

O método TCPModel proposto neste trabalho, foi concebido com base nessa afirmativa. Para detectar esses desvios no comportamento do protocolo TCP, o método TCPModel implementa um algoritmo denominado Threshold Adaptativo que é responsável por analisar se a relação entre pacotes recebidos e enviados, durante um intervalo de tempo, excede certo limiar.

A relação é obtida através da observação dos pacotes TCP trocados entre clientes e servidores, agrupados na forma de *socks*. Segundo Stevens [31], um *sock* é representado pelo agrupamento de fluxos que têm em comum o endereço IP e a porta do servidor, como mostra a Figura 4-2.

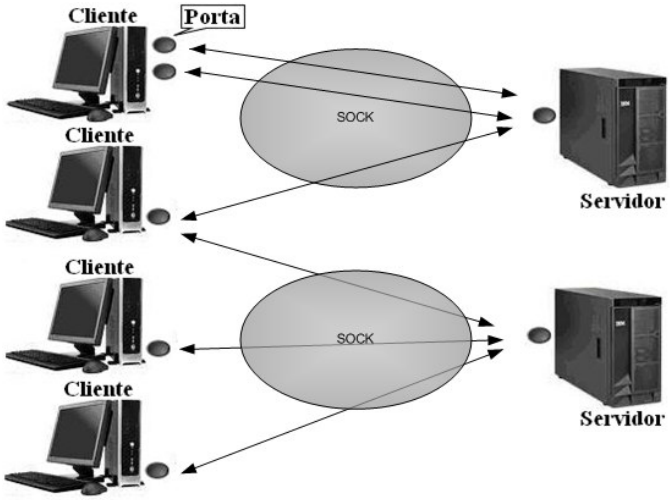

Figura 4-2: Representação de um Sock.

A seguir é descrito o funcionamento do algoritmo em detalhes.

Threshold Adaptativo.

Se considerarmos x_n como a relação entre pacotes enviados e recebidos num certo intervalo *n* para um dado *sock*, e μ_{n-1} a estimativa média ou valor esperado dessa relação no intervalo anterior, então, o fórmula que verifica se o limiar foi excedido e dada pela Equação 1.

$$x_n \geq (\alpha + 1)\,\bar{\mu}_{n-1}$$

Equação 1: Verificação do limiar para o algoritmo Threshold Adaptativo

Na Equação 1, α representa o percentual que deve ser aplicado ao valor esperado para estabelecer o limiar onde a taxa seria entendida como indicação de ataque.

A aplicação direta da equação apresentada acima, apesar de responder rápido a eventos da rede, pode levar a um grau elevado de falsos positivos. Na tentativa de amenizar este problema a

condição de sinalização do algoritmo em questão representa uma variação da Equação 1, que leva em consideração a quantidade de vezes consecutiva em que o limiar é superado. Para tanto é definida uma função apresentada abaixo.

$$f(x) = \begin{cases} 1 & x \geq (\alpha + 1)\,\bar{\mu}_{n-1} \\ 0 & x < (\alpha + 1)\,\bar{\mu}_{n-1} \end{cases}$$

Equação 2: Função de indicação do Threshold Adaptativo.

A Equação 2 representa uma função de indicação que retorna 1 se a Equação 1 se verifica ou 0 caso contrário. A equação final de sinalização pode ser vista a seguir.

$$\sum_{i=n-k+1}^{n} f(x_i) \geq k$$

Equação 3: Teste de sinalização do Threshold Adaptativo.

Na Equação 3, k > 1 é um parâmetro que indica o número de intervalos consecutivos que serão necessários para disparar o alarme.

Fica claro que esta abordagem é fortemente dependente do valor parâmetro α e mais ainda do valor esperado μ_{n-1}. Levando em consideração que este valor está completamente relacionado às condições da rede, é preciso escolher uma forma de calcular um valor que seja representativo da rede analisada, sem contudo, sofrer com picos e oscilações repentinas de comportamento. O método Alisamento Exponencial Simples foi escolhido para calcular o valor de μ_{n-1}. Detalhes dessa técnica são apresentados a seguir.

Alisamento Exponencial Simples

Alisamento exponencial ou *EWMA - Exponentially Weighted Moving Average* - é uma técnica de tratamento de dados históricos (série temporal) que realiza a previsão do valor futuro de uma variável temporal através de suavizações (alisamento) de seus valores passados.

O Alisamento Exponencial Simples é entendido como uma média ponderada das observações realizadas e pode ser calculado através da equação apresentada a segui.

$$\bar{\mu}_n = \beta\,\bar{\mu}_{n-1} + (1-\beta)\,x_n$$

Equação 4: Cálculo do Alisamento Exponencial Simples.

Na Equação 4, $\overline{\mu}_n$ representa a previsão no tempo n, β é a constante de alisamento e x_n é o valor observado no tempo n.

A constante de alisamento β é o que determina o grau de relevância dos últimos dados amostrados. Um fator muito pequeno dá um peso excessivo à observação mais recente, amenizando os movimentos anteriores. Um fator muito grande terá o efeito contrário, fazendo com que as observações mais antigas tenham um peso maior. β pode assumir os valores de 0 a 1 e é entendido tanto como uma porcentagem ou intervalos de tempo. Por exemplo, um fator de 95% seria equivalente a β = 0.95. Para calcular β em função do número de intervalos de tempo, utiliza-se a X, por exemplo, para N = 9 é β = 0.2.

$$\beta = \frac{2}{(N+1)}$$

Equação 5: Cálculo de β através do intervalo de tempo N.

4.1.2.2 SIPModel

O segundo método de classificação implementado no ChkModel é denominado Source IP Model ou SIPModel. Este método foi concebido com base nas observações de traces de ataques realizadas por Jung [32]. Em seu trabalho, Jung mostrou que durante um ataque de DDoS apenas 0,6% a 14% dos endereços IP encontrados durante um ataque de DDoS mantiveram algum tipo de conexão com a vítima no passado. Em outras palavras, a maioria dos endereços IP de origem são novos à vitima. Além disso, apenas uma pequena porção de endereços IPs aparece como novo numa situação normal[33]. Em seu trabalho, Jung também argumenta que sobre o ponte de vista do aumento no volume de tráfego e requisições geradas a um serviço apesar, das características semelhantes entre ataques DDoS e eventos *flash crownd* [32], a maioria dos endereços IP de origem em eventos do tipo *flash crownd* já efetuaram requisições ao serviço em questão no passado diferentemente do que acontece com ataques DDoS.

Baseado na afirmação acima, o SIPModel foi então construído para detectar ataques de negação de serviço distribuído através do monitoramento do aumento na quantidade de novos endereços IP de origem que trafegam na rede.

O método proposto é composto por dois componentes: LIC e CUSUM. O primeiro é responsável por manter uma lista de IPs que acessaram a rede e que são confiáveis; o segundo

componente é o algoritmo de detecção responsável por periodicamente verificar se a porcentagem de novos endereços IPs não excede um determinado limiar.

LIC – Lista de IPs Confiáveis

O LIC é o componente responsável por manter uma lista com o histórico de boas conexões já estabelecidas na rede, mais especificamente, dos endereços IP. O LIC inicia com uma base já conhecida de IPs confiáveis e atualiza essa lista utilizando um algoritmo específico. O procedimento adotado pelo LIC é bastante simples: cada conexão deve ter no mínimo três pacotes e a razão entre pacotes enviados e recebidos não deve exceder a razão média da rede.

O LIC mantém duas tabelas: a primeira é responsável por armazenar todos os endereços IP de origem que acessaram a rede do período atual; a segunda guarda a lista de clientes legítimos. Comparando as duas tabelas é possível identificar a quantidade de novos IPs na rede no intervalo de tempo em questão.

As duas tabelas são compostas por dois campos, um para o endereço IP e outro para o tempo de chegada do último pacote para esse IP. A Figura 4-3 ilustra o formato das tabelas do LIC. A motivação para adicionar o campo tempo de chegada é evitar a que a lista de IPs cresça indefinidamente e contenha clientes que não mais acessam a rede. De posse da informação do tempo de chagada do último pacote é possível eliminar as entradas mais antigas da tabela, evitando assim, operar além da capacidade de armazenamento disponível.

Endereço IP de Origem	Tempo de Chegada
200.192.248.7	1200858097812
200.192.60.72	1200858257835
⋮	
192.168.0.60	1200858357849

Figura 4-3: Tabela de Clientes Legítimos

CUSUM

O algoritmo da soma acumulativa (CUSUM – Cumulative Sum) pertence à família de algoritmos que se baseiam em testes hipótese e são capazes de detectar pequenas mudanças na distribuição da variável em questão. É uma ferramenta estatística que acumula informações das amostras de um processo ponderando-as igualmente, ou seja, as amostras têm o mesmo peso.

O CUSUM baseia-se no fato de que, se uma mudança ocorre, a distribuição de probabilidade de uma seqüência aleatória também irá mudar. Geralmente, o CUSUM exige um modelo paramétrico para a seqüência aleatória de modo a que a função densidade de probabilidade possa ser aplicada para monitorar a seqüência. Infelizmente a Internet é uma entidade muito complicada e dinâmica, sendo a construção de modelos teóricos para modelá-la um problema em aberto. Portanto, o problema abordado neste trabalho requer a utilização do CUSUM sob um modelo não paramétrico [34], visto que métodos não paramétricos não são presos a modelos e assim tem uma aplicabilidade maior e são mais fáceis de serem utilizados em análises desse tipo. De forma geral, a técnica é baseada no modelo apresentado por [35] para detecção de ataques usando o CUSUM.

A Figura 4-4 abaixo ilustra um exemplo de comportamento do CUSUM para uma seqüência randômica $\{X_n\}$ em análise. O primeiro gráfico da Figura 4-4 aponta uma abrupta mudança na média no ponto m, de α até $\alpha + h$.

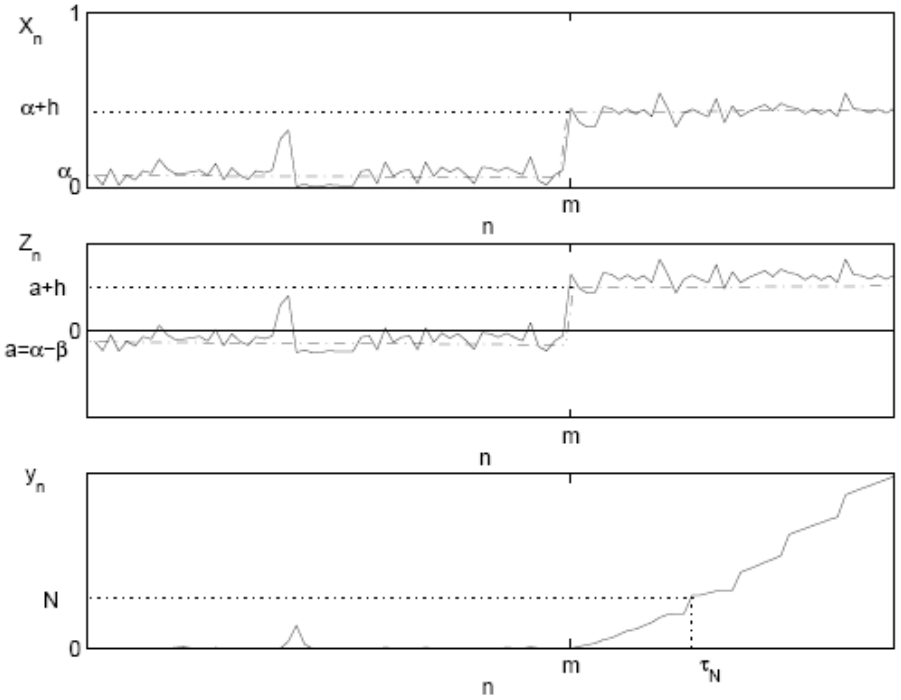

Figura 4-4: Gráfico de controle CUSUM

O valor de α representa a média dos valores de {Xn} para o intervalo de tempo considerado e h pode ser visto como a elevação mínima em {Xn} para que o comportamento seja considerado um ataque.

Para o SIPModel, $\{X_n\}$ representa o porcentual de novos endereços IP em um dado intervalo n. Em situações normais essa fração tem valor próximo de 0, isto é, $E(Xn) = \alpha << 1$, ou seja, a esperança de X_n é bem menor que 1.

Uma das condições de aplicação para o CUSUM não paramétrico é que os valores médios da seqüência aleatória sejam negativos sobre situação normal e positivo em situação de ataque. Desta forma $\{X_n\}$ precisa ser transformada em outra seqüência $\{Z_n\}$. O segundo gráfico da Figura 4-4 mostra o comportamento dessa nova seqüência $Z_n = X_n - \beta$. O parâmetro β é o responsável por garantir que os valores normais em $\{X_n\}$ sejam negativos em $\{Z_n\}$. Ainda observando o segundo gráfico da Figura 4-4, $a = \alpha - \beta$ representa a média da nova seqüência.

A idéia principal por trás do algoritmo não paramétrico CUSUM é que os valores de Xn que são muito maiores que o valor esperado vão acumulando até que um limiar seja atingido onde é feita uma sinalização. Para atingir esse objetivo a seqüência $\{Z_n\}$ passa por uma transformada que gera uma nova seqüência $Y_n = (Y_{n-1} + Z_n)+$, onde $Y_0 = 0$. O comportamento de $\{Y_n\}$, último gráfico da Figura 4-4, também é bastante simples, a seqüência não permite valores negativos e opera acumulando os valores positivos de $\{Z_n\}$.

Por fim a função de decisão, descrita na Equação 6 verifica se o valor yn ultrapassa o limiar estabelecido N.

$$d_N(y_n) = \begin{cases} 1 & y_n > N \\ 0 & y_n \leq N \end{cases}$$

Equação 6: Função decisão do CUSUM.

4.1.2.3 Base de Dados

Como visto anteriormente, o componente base de dados é o responsável por manter as informações necessárias a cada um dos métodos de classificação. São guardados dados referentes ao modelo do comportamento normal bem como as atualizações feitas no mesmo. Além disso, este componente pode guardar informações sobre as últimas classificações realizadas ou os próprios

eventos já observados, para futura análise. Uma funcionalidade muito importante da Base de Dados no sistema desenvolvido foi guardar a lista inicial de IPs do LIC e receber atualizações da lista pelo próprio LIC.

4.1.2.4 Resposta

O componente de resposta é responsável por receber as sinalizações vindas dos métodos de analise, processá-las e gerar uma resposta.

Apesar de ser possível implementar políticas elaboradas que levem em consideração alertas anteriores ou mesmo que atuem diretamente no controle da rede, neste trabalho o componente de resposta se restringiu a gerar um alerta sempre que um dos módulos do componente de analise sinalizasse a detecção de um ataque.

5 Estudos de Caso e Resultados

Esta seção descreve o processo de avaliação do ChkModel. O ambiente de teste utilizado, a ferramenta utilizada para gerar os ataques, as métricas utilizadas na avaliação e os resultados iniciais obtidos são apresentados em detalhes. Os experimentos foram realizados nos dias 16, 17 e 19 de Janeiro de 2008. Os resultados apresentados a seguir representam a média referente a 30 replicações para todos os experimentos com um intervalo de confiança de 95%.

5.1 Ambiente de Teste

Para realizar os testes do ChkModel foram utilizadas as instalações do GPRT (Grupo de Pesquisas em Redes e Telecomunicações) da Universidade Federal de Pernambuco (UFPE) visando criar um ambiente controlado que se assemelha ao real e capaz de permitir ataques DDoS. Foram utilizados 52 PCs desktops, 2 switches com 24 portas 10/100/1000 Mbps e 2 servidores (processador Athlon XP 4200+ 64bits, com 2Gbytes de RAM e 160Gbytes de HDD), um atuando como roteador/gateway/firewall e outro com a implementação do ChkModel. Windows XP, distribuições Linux (Gentoo, Ubuntu, Debian, Red Hat e Slackware) e FreeBSD foram os sistemas operacionais utilizados. Por razões operacionais, o tráfego da rede do GPRT foi espelhado para o servidor de análise. A Figura 5-1 apresenta a topologia da rede do ambiente de teste.

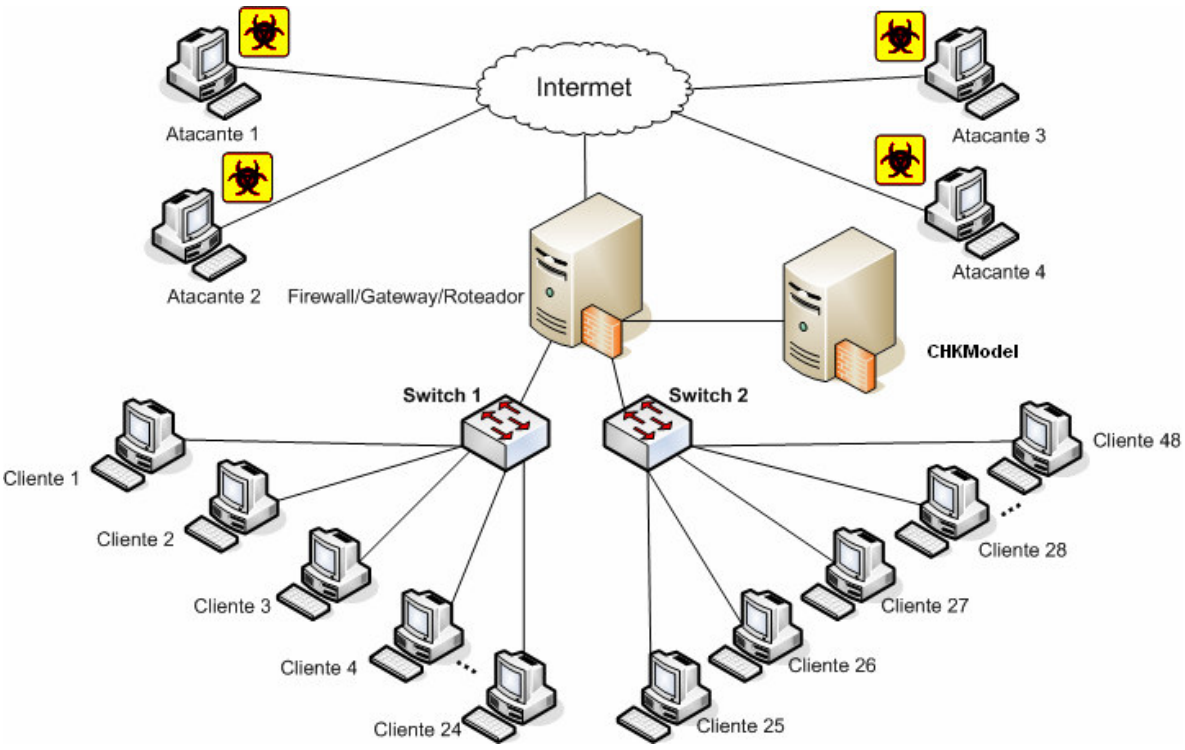

Figura 5-1. Topologia do Ambiente de Testes.

5.2 Tráfego de Ataque

Foram utilizados dois tipos diferentes de ataque, o SynFlood e o HTTP DoS. No primeiro caso, para geração do tráfego de ataque, foram utilizados 4 PCs executando um *script* que emprega a ferramenta Packit [36]. Após escolhida uma vítima, o script cria pacotes customizados com endereços IP de origem forjados. Cada pacote gerado com o script possui 1 Kbyte de tamanho, a *flag* TCP SYN ativada. Além disso, cada IP forjado gera 180 pacotes distribuídos uniformemente num intervalo de um minuto. A cada ataque são gerados de 15 a 20 novos IPs forjados, o que significa uma taxa média de 50 pacotes por segundo.

O segundo caso trata-se de um ataque de negação de serviço que funciona através de inundações de requisições HTTP (*HTTP Flood Denial of Service*)[37]. Para simular esse comportamento foi utilizada a ferramenta chamada *DoSHTTP v2.0[38]*. A ferramenta é capaz de utilizar múltiplos soquetes assíncronos para obter inundações efetivas. Apesar de não ser possível simular ataques de negação de serviço distribuídos com apenas uma máquina, é possível utilizar a ferramenta em várias máquinas para obter o comportamento esperado. Quatro máquinas atacantes foram utilizadas com esse propósito, operando com uma alta carga de requisições, ainda que de maneira pouco distribuída.

5.3 Métricas de Avaliação

Para avaliação do ChkModel foram utilizadas as seguintes métricas:

- **Probabilidade de Detecção** – mede a eficácia de detecção ataques. Entendida como a percentagem de ataques para os quais um alarme foi gerado.
- **Taxa de Falso Positivo** – Corresponde ao percentual de alarmes que não correspondem a um ataque.
- **Tempo de Detecção** – Intervalo de tempo entre o momento que um ataque é executado e um alarme é disparado.

5.4 Resultados

Esta seção apresenta os resultados das avaliações realizadas com o CHKModel. Os seguintes parâmetros foram utilizados:

- **TCPModel** – $k = 4$; $\mu_0 = 3$, $\alpha = 0.7$ e $\beta = 0.98$.
- **SIPModel** – $\mu_0 = 0.08$, EWMA = 0.98.

5.4.1 Ataque SynFlood

Os primeiros experimentos foram realizados considerando ataques do tipo SynFlood. O intervalo de tempo considerado no componente de análise, bem como para as análises realizadas foi de dez segundos. A Figura 5-2 mostra o a quantidade de pacotes legítimos e de ataque que trafegam na rede em cada intervalo de tempo observado. O eixo horizontal na figura representa o número de intervalos de tempo, onde 0 e 130 correspondem respectivamente ao primeiro intervalo e, aproximadamente 22 minutos depois, o último intervalo. A média de pacotes legítimos, assim como os de ataque, foi de 2000 pacotes por intervalo de tempo. Portanto, nos períodos de ataque, a vazão média da rede aumenta em 100%. A relação média de pacotes enviados por recebido para uma determinada máquina manteve-se aproximadamente em 1,5 quando em situação normal e variava de 3 a 8 quando o ataque era iniciado. A Figura 5-2 mostra também que foram realizados dez ataques com um minuto de duração e com igual intervalo entre eles.

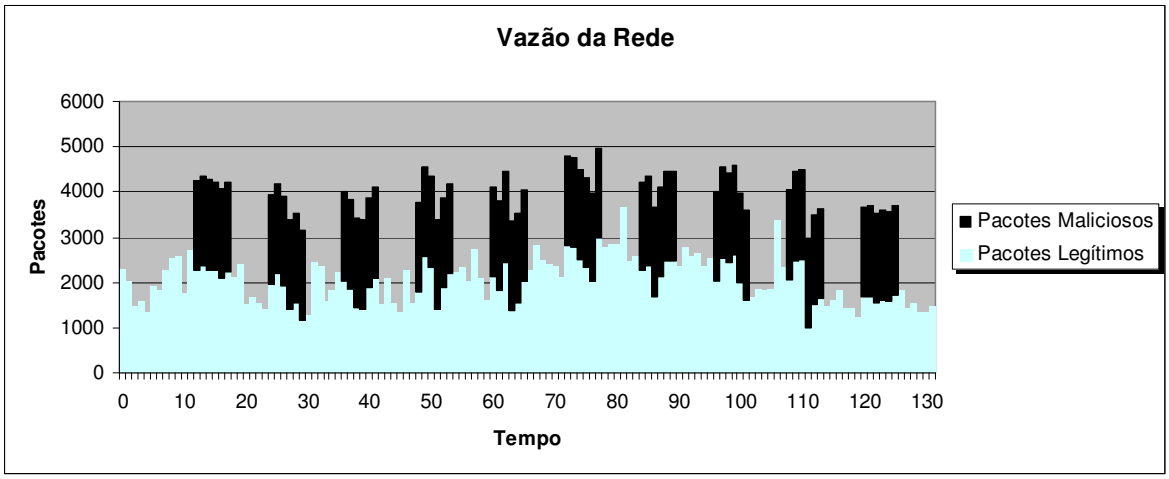

Figura 5-2: Vazão da rede para o cenário de ataque SynFlood.

A Figura 5-3 apresenta a quantidade de endereços IP que aparecem na rede a cada intervalo de tempo considerado (dez segundos). Os valores apontam que durante o estado normal da rede o número de IPs quase não ultrapassa 200, além disso, sobre estas condições o número de novos IPs não representa nem 9% do total. Durante o ataque a rede passa a receber em média 250 IPs por intervalo de tempo. Isto representa um acréscimo médio de 50 novos endereços IPs. Além disso, esses valores chegam a representar 30% da rede, um acréscimo de 21% quando comparado ao estado normal. Essa configuração do cenário simula o que ocorre com ataques de DDoS altamente distribuídos.

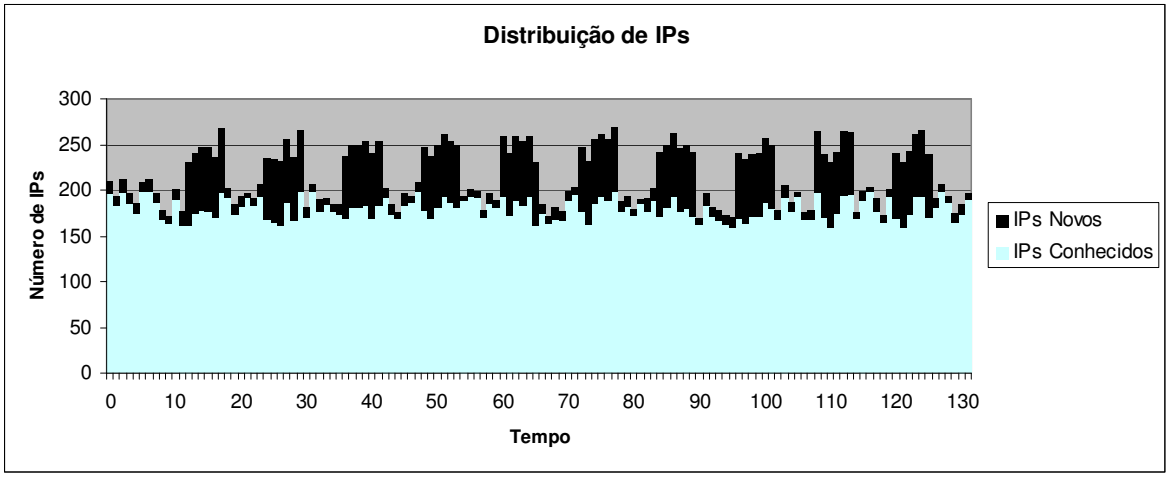

Figura 5-3: Distribuição do número de IPs no tempo para o cenário SynFlood.

A Figura 5-4 apresenta os momentos de ataque e onde o CHKModel gerou um alerta, representados por uma barra no eixo vertical. Neste cenário, o fato de muitos IPs forjados inundarem a rede, contribuiu para o bom desempenho do sistema. Todos os ataques foram detectados, correspondendo a uma taxa de detecção de 100%. A diferença razoável entre porcentagem média de novos IPs na rede durante o estado normal da rede e nos momentos de ataque levou a bons desempenhos também no tempo de detecção que foi de apenas 1,6 intervalos de tempo. O grande responsável então pelos bons resultados foi o método SIPModel. A melhora nos momentos finais, que levaram apenas um intervalo de tempo, é explicada pela adequação automática dos parâmetros ao estado da rede, que interferem de maneira decisiva do desempenho do sistema. Apesar de no início do desenvolvimento do sistema e com seus primeiros testes muitos alarmes falsos tenham sido gerados, quase todos podiam ser explicados pela escolha dos parâmetros iniciais dos métodos de análise. Um dos eventos que gerava uma alta taxa de falsos positivos estava relacionado à aplicações do tipo *peer-to-peer* como a aplicação BitTorrent. Quando programas desse tipo estavam operando na rede, muitas vezes, várias requisições não eram respondidas devido a algum usuário ter interrompido o compartilhamento, contudo a aplicação continuava tentando entrar em contato. Neste caso o TCPModel identificava uma possível sobrecarga no cliente em questão. Também foi possível observar que esse comportamento não se repetia por mais três vezes consecutivas, no intervalo considerado de dez segundos. O parâmetro "k" do TCPModel, que indica a quantidade de violações consecutivas necessárias, foi então ajustado para contemplar esse tipo de situação. Com o tempo, foi possível compreender razoavelmente a dinâmica da rede, com isso os parâmetros iniciais foram ajustados e nos testes finais, para este cenário, não foram detectados falsos positivos.

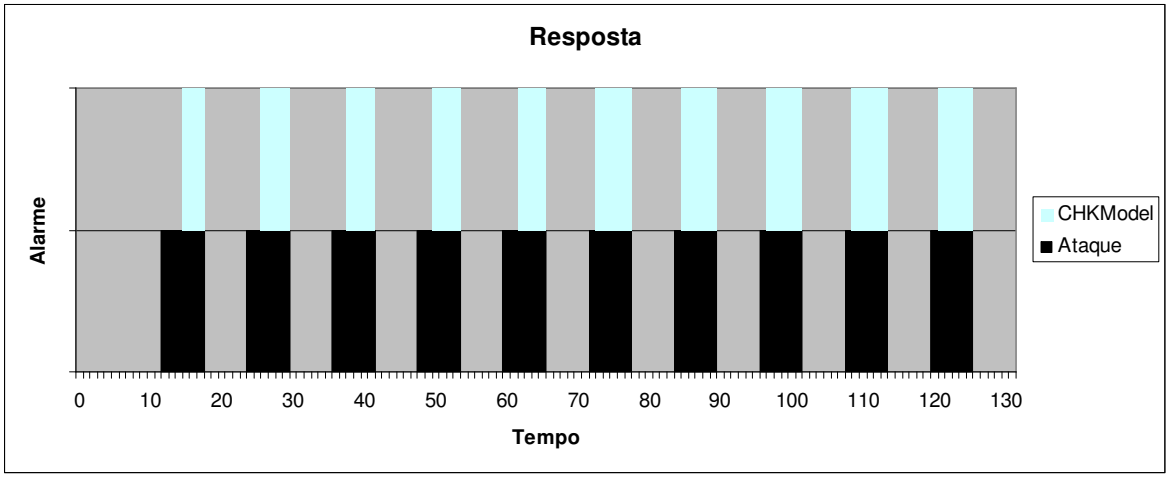

Figura 5-4: Sinalizações realizadas pelo ChkModel e ataques para o cenário SynFlood.

5.4.2 HTTP DoS

O segundo cenário avaliado corresponde aos ataques do tipo HTTP DoS. Assim como nos experimentos anteriores, o intervalo de tempo considerado nos testes foi de 10 segundos. A Figura 5-5 mostra a relação entre pacotes enviados durante os novos experimentos realizados. A média de pacotes legítimos aparece um pouco maior do que nos experimentos anteriores com aproximadamente 2 mil pacotes por intervalo de tempo. Uma diferença brutal está na quantidade de pacotes maliciosos, bem como sua relação com os pacotes legítimos. O ataque HTTP DoS representou um aumento médio de aproximadamente 8 mil pacotes. Portanto, nos períodos de ataque, a vazão média da rede aumenta em 300%, o triplo do observado no SynFlood. Neste cenário o valor da relação entre pacotes enviados por recebido manteve-se aproximadamente em 1,6 quando em situação normal e variava de 4 até 10 quando o ataque era iniciado. Novamente foram realizados 10 ataques com um minuto de duração e com igual intervalo entre eles.

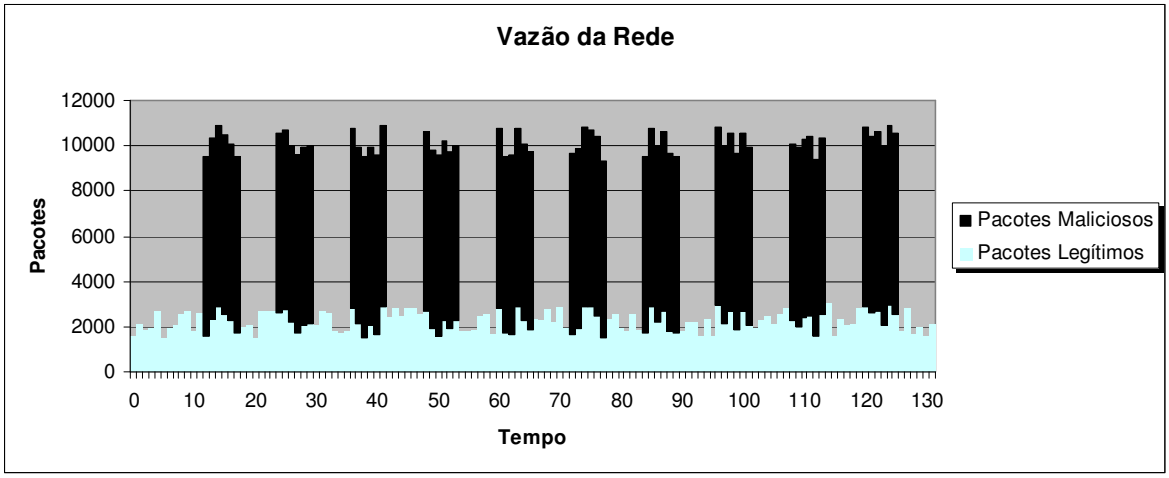

Figura 5-5: Vazão da rede para o cenário de ataque HTTP DoS.

Devido à característica pouco distribuída do ataque realizado neste segundo cenário, através da Figura 5-6 não é possível identificar os momentos onde são realizados os ataques. O tráfego maior no segundo cenário, observado na Figura 5-5 pode ser relacionado a um maior número endereços IP que trafegam na rede. Isto pode ser observado pelas várias vezes esse valor supera 200, algo que não ocorreu na rede nos durante os testes realizados com o cenário anterior. Apesar da taxa de novos endereços IP ultrapassar em alguns momentos 10%, a média foi de 6%, inclusive durante os ataques. Essa configuração não é favorável ao método SIPModel do componente de análise do CHKModel. Entretanto, a sobrecarga gerada na vítima do ataque pode ser identificada pelo TCPModel.

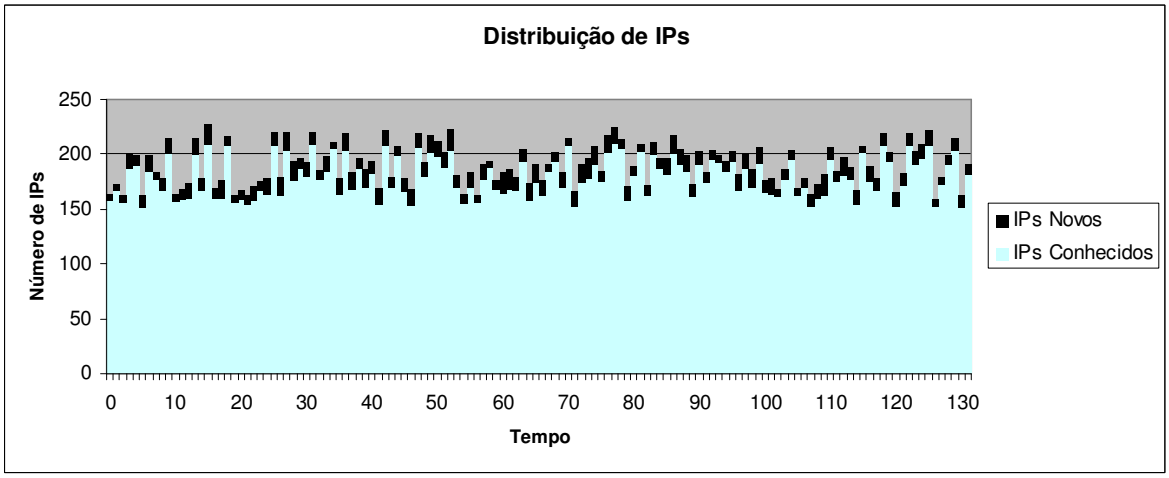

Figura 5-6: Distribuição do número de IPs no tempo para o cenário SynFlood.

A Figura 5-7 mostra a eficiência do CHKModel para o novo cenário realizado. Para o HTTP DoS a quantidade excessiva de requisições inválidas geraram muitas perdas ou sobrecarga na vítima. Este comportamento eleva a relação entre pacotes recebidos e enviados, favorecendo o método TCPModel. Novamente, todos os ataques foram detectados, correspondendo a uma taxa de detecção de 100%. O comportamento do TCPModel nas detecções é bem diferente do observado no SIPModel. Enquanto o primeiro, sempre que identifica um ataque leva "k" períodos para poder confirmá-lo, o segundo pode apresentar uma variação nesse período em função da taxa observada. Como o valor de "k" foi estabelecido em 4, em nenhuma análise será identificada um desempenho no tempo de detecção melhor que três intervalos. Novamente, a melhora nos momentos finais, pode ser explicada pela adequação dos parâmetros ao estado da rede. Neste caso, ao valor esperado da taxa entre pacotes recebidos e enviados. Uma questão poderia ser levantada relativa ao período de treinamento dos métodos que está relacionado com o desempenho inicial do sistema. O fato de serem realizados ataques prematuramente é uma tentativa de mostrar o tempo que o sistema leva para se estabilizar dado os parâmetros iniciais e as condições da rede. Para os cenários analisados não foram detectados falsos positivos. Contudo, é importante ressaltar que o desempenho do sistema nesse quesito é fortemente dependente do conhecimento prévio do comportamento da rede para ajuste dos parâmetros.

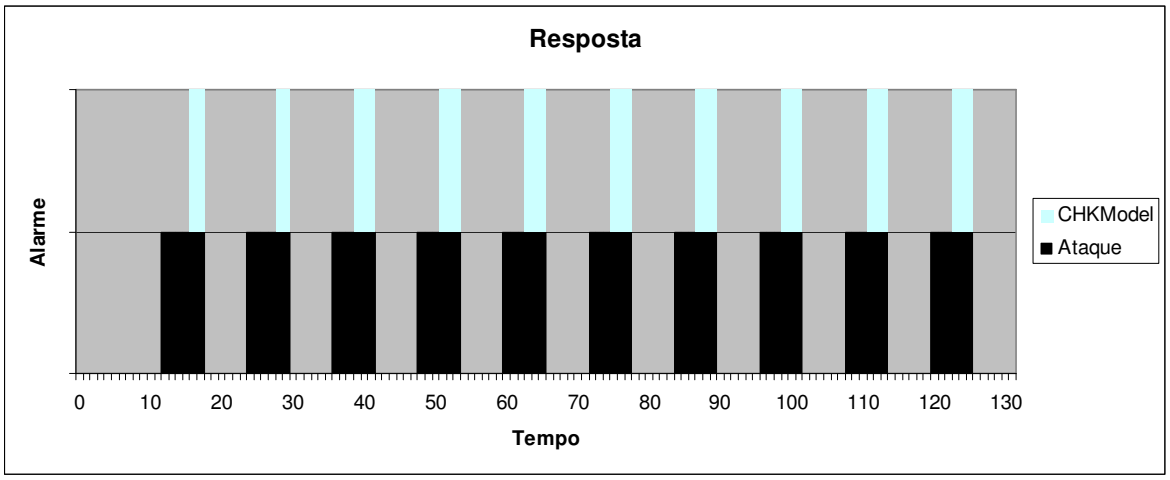

Figura 5-7: Sinalizações realizadas pelo ChkModel e ataques para o cenário HTTP DoS.

6 Conclusões e Trabalhos Futuros

Este trabalho apresentou uma nova abordagem de defesa contra ataques DDoS baseado na análise estatística da relação entre a quantidade de pacotes enviados e recebidos em um fluxo e no percentual de novos endereços IPs conectados a rede. Além disso, o trabalho investigou o desempenho para ataques com diferentes características mostrando que embora os métodos de detecção propostos no ChkModel sejam simples ambos apresentaram um bom desempenho para detecção de ataques DDoS do tipo SYN Flooding e HTTP DDoS.

Este trabalho apresenta várias oportunidades de pesquisa e pode ser estendido em várias direções:

1. Avaliar os métodos propostos em outros cenários com diferentes tipos de ataques;
2. Projetar e implementar novos métodos de detecção. Um trabalho interessante seria incluir uma avaliação estatística sobre o intervalo de chegada dos pacotes [1];
3. Projetar um mecanismo para executar as ações de eliminar ou mitigar os ataques. Este mecanismo deve ser capaz de interagir com diferentes elementos de rede como firewall e roteadores;
4. Desenvolver um esquema de cooperação mais elaborado entre os métodos propostos. O processo de tomada de decisão pode ser elaborado usando diferentes técnicas tais como árvores de decisão, esquemas de votação, redes neurais, entre outros.

Referências

[1] L. Feinstein e D. Schnackenberg. Statistical Approaches to DDoS Attack Detection and Response. Proceedings of the DARPA Information Survivability Conference and Expostion(DISCEX'03), Abril de 2003.

[2] C. Manikopoulos e S. Papavassiliou. "Network Intrusion and Fault Detection: A Statistical Anomaly Approach", IEEE Communications Magazine, Outubro de 2002.

[3] L. Feinstein e D. Schnackenberg. "DDoS Tolerant Network", Proceedings of the DARPA Information Survivability Conference and Expostion(DISCEX'03), Abril de 2003.

[4] NIST/SEMATECH, "e-Handbook of Statistical Methods". Disponível em http://www.itl.nist.gov/div898/handbook/, último acesso: 17/01/2008.

[5] C.C. ALVES, "Gráficos de controle CUSUM: um enfoque dinâmico para a análise estatística de procesos". Dissertação de Mestrado em Engenharia de Produção, UFSC, 2003.

[6] J.P. Anderson, "Computer security threat monitoring and surveillance", James P Anderson Co., Fort, Technical Report 98-17, Abril de 1980.

[7] A. Patcha e J.-M. Park, "An overview of anomaly detection techniques: existing solutions and latest technological trends," Elsevier Computer Networks, Vol. 51, Issue 12, 2007, pp. 3448–3470.

[8] J. Postel, "Transmission Control Protocol", RFC793, Stembro de 1981.

[9] Staniford-Chen S. and B. Tung, The Common Intrusion Detection Framework (CIDF)

[10] H. Richard e S. Mark. "The architecture of a network level intrusion detection system. Technical Report CS90-20. Department of Computer Science, University of New Mexico, 1990.

[11] S. Kumar e E.H. Spafford, "An application of pattern matching in intrusion detection", The COAST Project, Department of Computer Sciences, Purdue University, Technical Report CSD-TR-94-013, Junho de 1994.

[12] Computer Emergency Response Team, "Denial of Service Attacks". Disponível em http://www.cert.org/tech_tips/denial_of_service.html, último acesso: 17/01/2008.

[13] M. Handley, E. Rescorla, "Internet Denial-of-Service Considerations", RFC4732, Novembro de 2006

[14] CERT Incident NOTE IN-99-04, Disponível em: http://www.cert.org/incident_notes/IN-99-04.html, último acesso: 17/01/2008.

[15] BUSINESSWEEK ONLINE, "Cyber Crime", Fevereiro de 2000. Disponível em: http://www.businessweek.com/2000/00_08/b3669001.htm. Acessado em: 17/01/2008.

[16] L. Robert, "Web worm targets White House", CNET News.com, Julho 2001. Disponível em http://www.news.com/2100-1001-270272.html, último acesso: 17/01/2008.

[17] K. Fernanda, "SCO confirma ataque pelo MyDoom e oferece US$ 250 mil por criador", Folha Online. Disponível em http://www1.folha.uol.com.br/folha/informatica/ult124u15060.shtml, último acesso: 17/01/2008.

[18] D. Dittrich, "The DoS Project's "trinoo" distributed Denial of Service attack tool, Outubro de 1999. Disponível em: http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt, último acesso: 17/01/2008.

[19] D. Dittrich, "The Tribe Flood Network distributed denial of service attack tool", Outubro de 1999. Disponível em: http://staff.washington.edu/dittrich/misc/tfn.analysis, último acesso: 17/01/2008.

[20] D. David, "The stacheldraht distributed denial of service attack tool", Dezembro de 1999. Disponível em: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt, em 17/01/2008.

[21] D. Sven, L. Neil e D. David, "Analyzing Distributed Denial of Service Tools: The Shaft Case", USENIX Association, Proceedings of the 14th Systems Administration Conference (LISA 2000), Dezembro de 2000.

[22] Jason Barlow and Woody Thrower, Axent Security Team, "TFN2K - An Analysis", Março de 2000. Disponível em: http://www.securiteam.com/securitynews/5YP0G000FS.html, último acesso: 17/01/2008.
[23] Ellen Messmer, "New denial-of-service attack tool uses chat programs", CNN.com, Setembo 2000. Disponível em http://archives.cnn.com/2000/TECH/computing/09/06/fear.trinity.idg/index.html, último acesso: 17/01/2008.
[24] S. M. Bellovin, "ICMP Traceback Messages", Março 2000. Disponível em http://www.cs.columbia.edu/~smb/papers/draft-bellovin-itrace-00.txt, último acesso: 17/01/2008.
[25] J. Ioannidis e S. Bellovin, "Implementing pushback: Router-based defense against DDos attacks." In Proceedings of the Symposium on Network and Distributed System Security (NDSS-02), 2002.
[26] J. Mirkovic, G. Prier, e P. Reiher, "Attacking DDoS at the Source," Proceedings of 10th IEEE International Conference on Network Protocols, pp. 312-321, Novembro 2002.
[27] R. Thomas, B. Mark, T. Johnson e J. Croall, "NetBouncer: client-legitimacy-based high-performance DDoS filtering," DARPA Information Survivability Conference and Exposition, 2003. Proceedings Vol. 1, 22-24, pp. 14 – 25, Abri de 2003.
[28] D. Keromytis, V. Misra, e D. Rubenstein, "SOS: An Architecture For Mitigating DDoS Attacks", IEEE Journal on Selected Areas in Communications (JSAC), Janeiro de 2004.
[29] TCPDUMP 2006. Disponível em http://www.tcpdump.org/, último acesso: 17/01/2008.
[30] JPCAP 2007, versão 0.7. Disponível em http://netresearch.ics.uci.edu/kfujii/jpcap/doc, último acesso: 17/01/2008.
[31] W. R. Stevens, "UNIX Network Programming, vol. 1, Second Edition: Networking APIs: Sockets and XTI", Prentice Hall, 1998.
[32] J. Jung, B. Krishnamurthy e M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites", MIT Laboratory for Computer Science, 2002
[33] T. Peng, C. Leckie e K. Ramamohanarao, "Protection from distributed denial of service attacks using history-based IP filtering", Proceedings of IEEE ICC 2003, pp. 482-486.
[34] B.E. Brodsky, B.S Darkhovsky, "Nonparametric Methods in Change-Point Problems", Kluwer Academic Publishers, 1993.
[35] H. Wang, D. Zhang, e K. G. Shin, "Detecting SYN flooding attacks," Proceedings of IEEE Infocom'2002, Junho de 2002.
[36] Intrusense. (2006) "Packit – Network Injection and Cature". Disponível em: www.intrusense.com/software/packit/, último acesso: 17/01/2008.
[37] Y. Takeshi, I. Takamasa e S. Iwao, "Detection of HTTP-GET flood attack bsed on analysis of page access behavior", IEEE Pacific Rim Conference on Communiocations, Computers and Signal Processing(PACRIM2007), pp.232-235, Agosto de 2007.
[38] HTTP DoS Test Tool (DoSHTTP 2.0). Disponível em http://www.ocketsoft.net, último acesso: 17/01/2008.