MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

RODOLFO SOARES ALVES DANTAS

DIFERENCIAÇÃO DE ATAQUES DDOS E FLASH CROWDS

Rio de Janeiro
2013

INSTITUTO MILITAR DE ENGENHARIA

RODOLFO SOARES ALVES DANTAS

DIFERENCIAÇÃO DE ATAQUES DDOS E FLASH CROWDS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Ronaldo Moreira Salles - Ph.D.
Co-orientador: Prof. Artur Ziviani - Dr.

Rio de Janeiro
2013

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

004.678 Dantas, Rodolfo Soares Alves
D192d Diferenciação de Ataques DDoS e Flash Crowds/ Rodolfo Soares Alves Dantas; orientado por Ronaldo Moreira Salles. – Rio de Janeiro: Instituto Militar de Engenharia, 2013.

74 p.: il., graf., tab.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2013.

1. Sistemas e Computação. 2. Segurança da Web. 3. Ataques DDoS. 4. Flash Crowds. I. Salles, Ronaldo Moreira. II. Título. III. Instituto Militar de Engenharia.

CDD 004.678

INSTITUTO MILITAR DE ENGENHARIA

RODOLFO SOARES ALVES DANTAS

DIFERENCIAÇÃO DE ATAQUES DDOS E FLASH CROWDS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Ronaldo Moreira Salles - Ph.D.

Co-orientador: Prof. Artur Ziviani - Dr.

Aprovada em 21 de maio de 2013 pela seguinte Banca Examinadora:

Prof. Ronaldo Moreira Salles - Ph.D. do IME - Presidente

Prof. Artur Ziviani - Dr. do LNCC

Prof. Anderson Fernandes P. dos Santos, D.Sc. do IME

Prof. Sidney Cunha de Lucena, D.Sc. da UNIRIO

Rio de Janeiro
2013

Dedico esta dissertação à Lívia dos Santos Dantas, minha filha.

AGRADECIMENTOS

Primeiramente à Deus por me dar saúde e força para vencer esse desafio.

À minha esposa Tamiris por todo apoio e paciência. Obrigado pelo carinho, amor e companheirismo. Sem o seu incentivo eu não teria conseguido.

Aos meus pais, Laurene e Ricardo, que sempre me apoiaram em todos os momentos da minha vida e não mediram esforços para a minha educação.

Aos meus orientadores, Artur Ziviani e Ronaldo Salles, por todos os ensinamentos, pela paciência e pela amizade. Muito obrigado.

Aos professores Antônio Tadeu e Wallace Pinheiro por toda ajuda no decorrer desta dissertação.

Aos meus amigos do mestrado pelo apoio e por todos os ensinamentos durante o curso. Vocês fazem parte dessa conquista.

À Capes pelo apoio financeiro durante o período de realização deste mestrado.

Por fim, à todos os professores e funcionários da Seção de Engenharia de Sistemas (SE/8) do Instituto Militar de Engenharia.

Rodolfo Soares Alves Dantas

"No meio da dificuldade encontra-se a oportunidade."
(Albert Einstein)

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABREVIATURAS

AR	-	*Aprendizado por Reforço*
CAPTCHA	-	*Completely Automated Public Turing test to tell Computers and Humans Apart*
CDN	-	*Content Delivery Network*
CPU	-	*Central Processing Unit*
DDoS	-	*Distributed Denial of Service*
DNS	-	*Domain Name System*
EOS	-	*Escape-Onsight*
HTML	-	*HyperText Markup Language*
HTTP	-	*HyperText Tranfer Protocol*
ICMP	-	*Internet Control Message Protocol*
IME	-	*Instituto Militar de Engenharia*
IP	-	*Internet Protocol*
I/O	-	*Input/Output*
MTV	-	*Model Template View*
RAM	-	*Random Access Memory*
SGDB	-	*Sistema de Gerenciamento de Banco de Dados*
SO	-	*Sistema Operacional*
SYN	-	*Synchronize*
TCP	-	*Transmission Control Protocol*
UDP	-	*User Datagram Protocol*
URL	-	*Uniform Resource Locator*

RESUMO

Ataques distribuídos de negação de serviço (DDoS) e fenômenos de *Flash Crowd* são duas grandes preocupações para a estabilidade e segurança da Web. Fenômenos de *Flash Crowd* são grandes quantidades de acessos legítimos aos sites da Web, enquanto ataques DDoS Web são pedidos maliciosos em grande volume, cujo objetivo é subverter o funcionamento normal do site impedindo o acesso de usuários legítimos. Dado que ambas as situações são, à primeira vista, caracterizadas por um aumento no volume de tráfego da rede direcionado a um ponto, diferenciar ataques DDoS Web e *Flash Crowds* é um desafio.

Nesse trabalho, é apresentada uma proposta para a diferenciação de ataques DDoS Web e fenômenos de *Flash Crowds* através da análise integrada das características dos sites-alvo e do tráfego de rede. Também é apresentada uma técnica de aprendizado por reforço para que haja adaptação às mudanças de acordo com os erros e acertos descobertos na detecção das anomalias. No trabalho, é utilizada uma autenticação gráfica para distinguir quais os acessos são provenientes de usuários humanos e quais acessos são feitos por robôs. Para a implementação da proposta, foi criada uma arquitetura conceitual baseada em camadas e uma aplicação chamada StopBots. A técnica proposta é avaliada através de alguns experimentos em um ambiente real de comércio eletrônico.

ABSTRACT

Distributed Denial of Service (DDoS) attacks and Flash Crowds are two concerns for the stability and safety of the Web services. Flash Crowds are characterized by a high volume of legitimate traffic while DDoS attacks are illegitimate flows arriving in large volume that aim to subvert the normal operation of the site. Both situations are characterized by a network traffic volume increase directed to a point and is a challenge distinguish DDoS attacks and Flash Crowds.

In this paper, we present a proposal for differentiating DDoS Web attacks and Flash Crowds through integrated analysis of the Web sites characteristics and network traffic. Also it is presented a reinforcement learning technique to make happen an adaptation to changes accordingly with mistakes and successes found out during the anomalies detection. Graphical tests are used to distinguish where the access are made from human users and which accesses are made by bots. It is created a conceptual architecture based on layers and an application called StopBots to implement the proposal. The proposed technique is evaluated through some experiments in a real e-commerce environment.

1 INTRODUÇÃO

Ao longo do tempo, a Internet vem se consolidando como uma das principais ferramentas de comunicação no mundo. Sua aplicabilidade abrange todos os aspectos da vida humana e grandes quantidades de dados são trocados diariamente através da Web. Com toda a popularidade que a Internet vem ganhando, seu grande uso traz consigo alguns problemas que precisam ser tratados especificamente. Um problema muito comum são os ataques distribuídos de negação de serviço - DDoS (*Distributed Denial of Service*) (MIRKOVIC, 2004). Esses ataques consistem nas tentativas de impedir que usuários comuns utilizem determinados serviços de um computador ou de um grupo de computadores através de técnicas que sobrecarregam esses computadores, a tal ponto que os usuários legítimos não consigam utilizá-los. Por outro lado, como a Internet é um meio extremamente dinâmico, existem situações em que alguns desses computadores podem sofrer acessos concentrados em determinadas ocasiões. Esse fenômeno é conhecido pelo nome de *Flash Crowd* (HONG, 2003) e deve-se a diversos fatores, que vão desde o anúncio de um determinado serviço na Web até a relevância que o serviço pode ter em uma determinada ocasião.

Por sua vez, em computação em nuvem, os provedores de acesso oferecem aos consumidores dois planos de fornecimento de recursos computacionais: instâncias[1] reservadas e instâncias sob-demanda. Em geral, o custo de utilização dos recursos computacionais aprovisionados pelo plano de instâncias reservadas é mais barato do que pelo plano sob-demanda, pois o consumidor tem de pagar ao fornecedor com antecedência. A Figura 1.1 mostra a comparação de preços de instâncias reservadas e instâncias sob-demanda do provedor Amazon (MURTY, 2008). Em azul, os valores de instâncias sob-demanda (*on-demand*) e em vermelho os valores de instâncias reservadas (*reserved instances*).

Com o plano de instâncias reservadas, o consumidor pode reduzir o custo total de recursos utilizados. No entanto, a decisão da melhor quantidade de recursos reservados é difícil de ser alcançada devido à incerteza da demanda e aos preços dos provedores de acesso. Além disso, os recursos computacionais deverão atender a demanda de acessos,

[1]Instâncias são servidores virtuais que podem ser inicializados em minutos, permitindo uma rápida escala de capacidade, para mais e para menos, à medida que os requisitos de computação forem alterados.

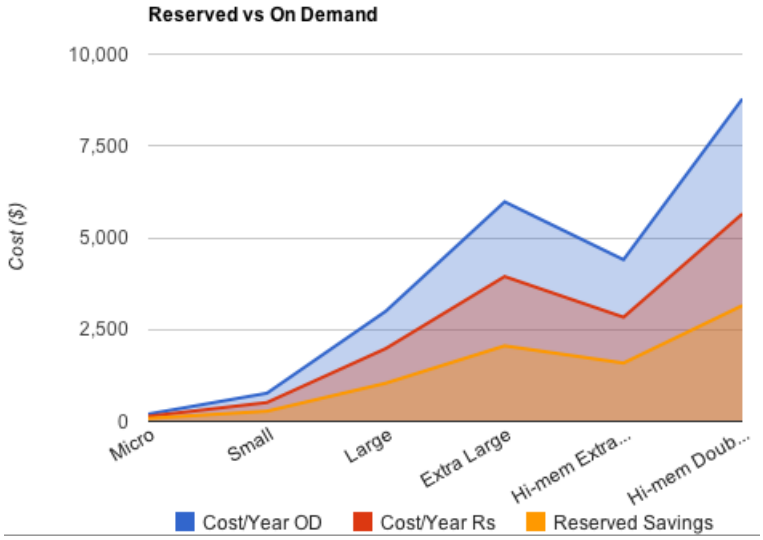

FIG. 1.1: Instâncias reservadas x sob-demanda.

cujos valores podem variar em duas situações específicas: ataques DDoS e fenômenos de *Flash Crowd*.

Para resolver esse problema, mecanismos precisam ser elaborados para detectar a ocorrência dessas anomalias e distingui-las com precisão. Enquanto os ataques DDoS deverão ser bloqueados, os fenômenos de *Flash Crowd* precisarão ser atendidos com o aumento de recursos computacionais. Dessa maneira, os consumidores da nuvem poderão minimizar o custo total de aprovisionamento de recursos em ambientes de computação em nuvem e utilizar a elasticidade da nuvem para prover recursos computacionais de acordo com a demanda, sem arcarem com o custo computacional dos acessos maliciosos.

1.1 ATAQUES DDOS

Ataques distribuídos de negação de serviço são um dos tipos mais sofisticados e populares de ataques virtuais. Através de *botnets*, por exemplo, esses ataques podem tirar um *site* do ar em até poucos minutos. As *botnets* são um conjunto de computadores infectados e controlados por *hackers* que funcionam como uma poderosa ferramenta para disseminar vírus, gerar *spam*, e, principalmente, efetuar ataques distribuídos de negação de serviço (FENG, 2011). Esses computadores contaminados e controlados remotamente também são conhecidos como *bots* ou zumbis. Atualmente, os ataques distribuídos de negação de serviço ocorrem quando usuários são convencidos a atuar em prol de uma causa em comum e utilizam redes com dezenas de milhares de computadores contaminados para

atacar serviços que estão disponíveis na Web (BUSSCHERS, 2012).

Apesar de possuírem o mesmo objetivo, os ataques DDoS (LAU, 2000) podem ser classificados em alguns tipos, conforme mostrado a seguir:

- **UDP *Flood***: O UDP *Flood* consiste no envio em larga escala de pacotes à portas aleatórias de um servidor atacado. Desta maneira, tendo que responder a uma grande quantidade de requisições, o servidor se torna inacessível para usuários legítimos.

- **ICMP *Flood***: Os ataques através de requisições ICMP são semelhantes aos ataques UDP e possuem a intenção de inundar o alvo atacado com pacotes ICMP, deixando-o indisponível.

- **TCP SYN *Flood***: O TCP SYN *Flood* atua com a intenção de sobrecarregar o alvo atacado através de solicitações sucessivas de conexão. O mecanismo consiste em explorar o processo de estabelecimento de conexão em três vias (*three-way handshake*) do protocolo TCP.

- **HTTP GET *Flood***: Os ataques HTTP GET *Flood* utilizam mecanismos que estabelecem uma conexão TCP regularmente, realizando o *three-way handshake* do protocolo TCP, e sua ação ocorre na camada de apresentação, através de inúmeras solicitações de conteúdo ao servidor Web.

Para dificultar a detecção, os atacantes estão deixando de inundar a largura de banda dos *sites*-alvo e preferindo os ataques que imitam o comportamento da navegação Web, tendo como alvo a camada superior de recursos, tais como CPU, memória e largura de banda dos servidores. Como resultado desses ataques, existe a dificuldade de defesa usando técnicas convencionais, pois as requisições maliciosas não se diferenciam das requisições legítimas, apenas na intenção dos acessos. Estes ataques são denominados como ataques DDoS Web e, embora essa expressão não seja considerada oficial, ela será empregada frequentemente nesse trabalho para indicar esses tipos de ataque.

1.1.1 ATAQUES DDOS WEB

Ataques à serviços Web podem ser classificados em dois tipos: os ataques que exploram uma vulnerabilidade de um serviço e os ataques que exploram a limitação de um serviço no provimento de um determinado recurso. Esse segundo tipo, que é conhecido por DDoS

Web, utiliza acessos maliciosos em grande escala para indisponibilizar um serviço Web por falta de recursos do servidor. Os ataques DDoS Web podem ser de três tipos: (i) quando a mesma página é requisitada várias vezes, (ii) quando várias páginas do mesmo *site* são requisitadas aleatoriamente e (iii) quando as páginas são acessadas buscando um padrão de navegação que uma pessoa utilizaria se estivesse navegando no *site*.

No primeiro caso, mostrado na Figura 1.2, uma análise mais profunda dos registros de acesso ao servidor mostraria que uma determinada página estaria sofrendo acessos repetidos com o mesmo padrão, e facilmente o ataque seria detectado, mesmo se o *site* sofresse um *Flash Crowd*, já que acessos legítimos tendem a navegar mais amplamente pelo *site*.

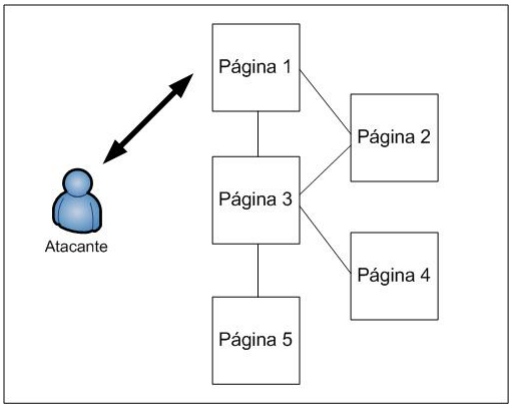

FIG. 1.2: Ataque constante em uma página.

No segundo caso, apesar do atacante utilizar páginas aleatórias, fatores como o padrão dos acessos às páginas e a repetição desse padrão seriam facilitadores na detecção do ataque. Esse tipo de ataque é mostrado na Figura 1.3.

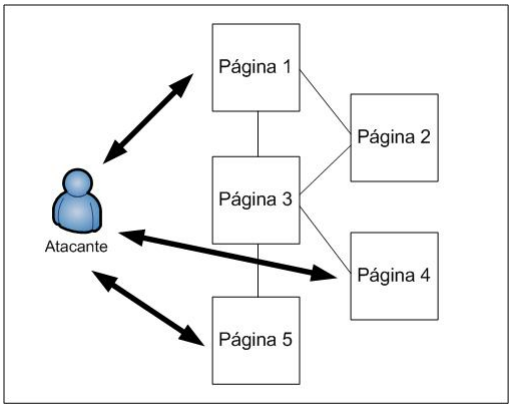

FIG. 1.3: Ataque em páginas aleatórias.

Por último, no terceiro caso, a diferenciação das anomalias se torna mais desafiadora, visto que o atacante utiliza padrões de navegação para imitar os fenômenos de *Flash Crowd* e a tarefa de diferenciar uma anomalia da outra se torna mais complicada. A demonstração desse ataque pode ser vista na Figura 1.4, onde o atacante começa na página 1, seguindo um atalho que o envia para a página 2 e depois para a página 3. As linhas em vermelho e em azul demonstram esse padrão de navegação.

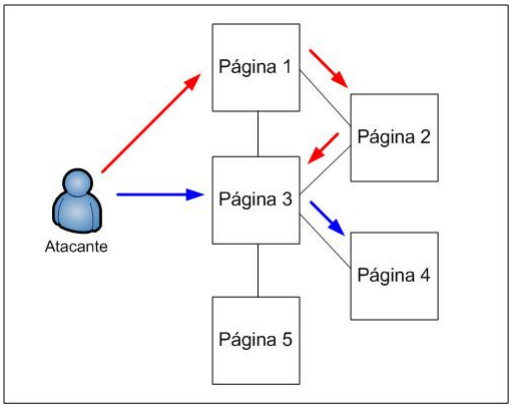

FIG. 1.4: Ataque com padrões de navegação.

1.2 FENÔMENOS DE *FLASH CROWDS*

O termo *Flash Crowd* foi comentado pela primeira vez em 1971, em uma história de ficção científica (NIVEN, 1973) referenciando uma situação onde milhares de pessoas voltaram no tempo para ver os acontecimentos históricos novamente. Na Internet, o fácil acesso à navegação Web e a rápida disseminação de notícias sobre um evento nos leva a uma situação semelhante, quando um número muito grande de usuários acessam simultaneamente um *site* popular. Os exemplos mais comuns de eventos incluem acontecimentos como a liberação do envio do imposto de renda, *webcasts* populares como o de grandes veículos de comunicação, eventos esportivos como os Jogos Olímpicos e a Copa do Mundo ou até mesmo uma liquidação que ocorre em *sites* de vendas *online*. Em alguns casos, a informação sobre a ocorrência dos eventos é conhecida de antemão. No entanto, existem muitos fenômenos de *Flash Crowd* que acontecem sem aviso prévio. Muitas vezes isso é devido a um evento catastrófico, como o ataque terrorista em 11 de setembro de 2001 nos Estados Unidos. *Sites* de notícias populares, como o da CNN por exemplo, sofreram um aumento dramático no número de pedidos e se tornaram indisponíveis devido a isto.

Ao contrário de ficção científica, o efeito de *Flash Crowds* nos servidores que hospedam

sites Web ou na infraestrutura de rede da Internet é real e pode ser devastador. Congestionamentos na camada de rede podem até impedir que alguns pedidos alcancem os servidores Web, e tornando-se real, podem causar atrasos significativos devido a perda de pacotes e tentativas de retransmissão. Já quando chegam aos servidores, eles por sua vez ficam incapazes de lidar com o volume de pedidos que precisam atender. Sejam quaisquer um dos motivos, os usuários que estão tentando obter informações durante um fenômeno de *Flash Crowd* são muitas vezes frustrados devido aos longos atrasos ou falhas definitivas. Em alguns casos, o servidor Web pode ficar indisponível devido aos efeitos de um *Flash Crowd*.

1.3 CONTRIBUIÇÃO

Tanto os fenômenos de *Flash Crowd* quanto os ataques DDoS Web são duas grandes preocupações para a estabilidade e segurança dos *sites* da Web e precisam ser tratados de maneira rápida para que não haja indisponibilidade do *site* vítima. Para isto, essas anomalias precisam ser abordadas diferentemente do lado da infraestrutura do *site*, já que acessos legítimos devem ser permitidos e acessos mal-intencionados devem ser bloqueados. Um dos problemas percebidos nesse contexto é a diferenciação de um ataque DDoS Web a um fenômeno de *Flash Crowd*, visto que ambos têm características muito parecidas. Como essas anomalias vêm se equiparando ao longo do tempo, é um desafio distinguir explicitamente os ataques DDoS dos fenômenos de *Flash Crowd*. Dada a imprevisibilidade e o aumento na frequência de *Flash Crowds* e a facilidade de elaboração de um ataque DDoS Web por meio de *botnets*, é importante que exista um mecanismo que consiga diferenciar e até mesmo mitigar os efeitos dessas anomalias.

Neste trabalho, é apresentada uma proposta para diferenciar os ataques DDoS Web e *Flash Crowds* através da análise integrada das características dos *sites* atacados e do tráfego da rede. O emprego de uma análise integrada visa melhorar a probabilidade na detecção e diferenciação dessas anomalias. O alvo da detecção serão *sites* que passam por fenômenos de *Flash Crowd* e podem ser vítimas de ataques DDoS em momentos alternados ou mesmo simultaneamente. Também é apresentada uma técnica de aprendizado por reforço para que o sistema se adapte às mudanças necessárias de acordo com os erros e acertos descobertos na detecção das anomalias. Na proposta, é utilizada uma autenticação gráfica para distinguir quais acessos são provenientes de usuários humanos e quais acessos são feitos por robôs. Os endereços IP que ignorarem os testes e inundarem o servidor

Web com requisições repetidas serão identificados. Essas máquinas serão classificadas como maliciosas, pois possuem a intenção de congestionar o servidor Web. Uma vez que essas máquinas são identificadas, os acessos provenientes delas são bloqueados durante um tempo, até que a carga de trabalho do servidor Web volte ao normal. Quando isto acontecer, a autenticação gráfica é desativada, liberando o acesso aos usuários legítimos que não são capazes ou não querem responder os testes. O desenvolvimento da proposta foi feito com a implementação de uma aplicação denominada StopBots, que tem por objetivo ser o centro da arquitetura e que possua toda a inteligência necessária ao sistema. Como resultado dessa técnica, procura-se melhorar a distinção de tais adventos, independente do fato da sobrecarga do servidor ser causada por um ataque de DDoS Web ou um fenômeno de *Flash Crowd*.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho encontra-se organizado da seguinte forma:

O Capítulo 2 apresenta os trabalhos relacionados ao problema estudado e suas principais características.

O Capítulo 3 descreve a metodologia proposta para a diferenciação dos fenômenos de *Flash Crowds* e ataques DDoS Web.

O Capítulo 4 detalha a arquitetura conceitual e sua implementação, bem como as ferramentas utilizadas para viabilizar o desenvolvimento da solução proposta.

O Capítulo 5 demonstra os resultados obtidos com a utilização da proposta. Para isso, são apresentadas as características do *site* de vendas *online* utilizado na implementação da proposta e demonstrada a metodologia empregada para a realização dos experimentos. Posteriormente são discutidos os resultados obtidos.

Finalmente, no Capítulo 6, são realizadas as considerações finais sobre a dissertação juntamente com os trabalhos futuros.

2 TRABALHOS RELACIONADOS

Existem alguns métodos já estudados para detecção e diferenciação de ataques DDoS e fenômenos de *Flash Crowd*, porém, com os avanços e a sofisticação das técnicas de ataque, há uma grande dificuldade de distingui-los com precisão e propostas para identificar essas duas anomalias são cada vez mais importantes. O trabalho de (JUNG, 2002) realizou a caracterização desses dois fenômenos. Essa pesquisa baseou-se no estudo das propriedades de ambos os tipos de anomalias com atenção especial às métricas e às características principais que as distinguem. Entre essas características, foi utilizado o padrão de tráfego de rede, a distribuição dos clientes, a média do número de requisições por cliente e os padrões de acesso aos arquivos durante um *Flash Crowd*. Após a identificação destas características, foram elaboradas técnicas que permitiram a formulação de uma estratégia para que os *sites* da Web possam descartar rapidamente as solicitações maliciosas oriundas de ataques DDoS. O trabalho também indicou que algumas redes de distribuição de conteúdo (*Content Delivery Networks* ou CDNs) não podem fornecer o nível de proteção desejado para os *sites* da Web contra fenômenos de *Flash Crowd* e propôs, portanto, uma melhor proteção das CDNs utilizando técnicas adaptativas à distribuição dos clientes e à resolução de nomes na Internet utilizando um servidor DNS personalizado.

Em outro trabalho, (LE, 2007) também se concentrou na análise e criação de métodos para reconhecimento de ataques DDoS. Foi percebido que, apesar dos ataques DDoS imitarem os fenômenos de *Flash Crowd*, algumas características são distintas entre eles e podem ser usadas para diferenciá-los. Um ataque pode disfarçar alguma dessas características, porém não pode ocultar simultaneamente todas as características estudadas. Entre elas, as características da origem das solicitações e dos endereços IP e a distribuição dos usuários da Web, por exemplo, é comum para a maioria dos fenômenos de *Flash Crowd*, porém um ataque DDoS dificilmente consegue reproduzir todas essas características, mesmo em um ataque mais sofisticado.

O trabalho de (LI, 2009) propôs o uso de métricas híbridas de probabilidade para detectar ataques DDoS e, através de experimentações e simulações, procurou mostrar que a métrica proposta pode não só detectar ataques DDoS dos fluxos normais, mas também pode distinguir os ataques DDoS de *Flash Crowds*. Além disso, este trabalho propõe a

redução da taxa de falso-positivos e da taxa de falso-negativos, utilizando métricas mais precisas, tentando melhorar a sensibilidade de detecção dessas anomalias.

Já o trabalho de (YU, 2009) pretendeu diferenciar os ataques DDoS de fenômenos de *Flash Crowd* motivados pelo seguinte fato: os fluxos de ataque são gerados pelos mesmos programas (ferramentas de ataque), no entanto, *Flash Crowds* são usuários distribuídos aleatoriamente em toda a Internet. Portanto, a semelhança entre os fluxos de ataques DDoS é muito maior do que fluxos de *Flash Crowds*. Os autores utilizaram métricas de distância, como a distância de Jeffrey, a distância Sibson, e a distância de Hellinger e, após comparadas as três métricas, identificou-se que a distância de Sibson é a mais adequada para essa finalidade. O algoritmo foi aplicado para conjuntos de dados reais e os resultados indicaram que o algoritmo proposto pode diferenciá-los com uma precisão em torno de 65%.

Em outro trabalho, (OIKONOMOU, 2009) propôs a diferenciação das anomalias através da modelagem do comportamento humano, que diferencia a navegação de robôs que efetuam ataques DDoS de usuários humanos. Eles propuseram uma abordagem transparente, modelando três aspectos do comportamento humano: (i) a dinâmica das requisições, observando o tempo de interação do usuário com o servidor Web no período de navegação no *site*, (ii) a semântica das requisições, observando a sequência das solicitações humanas e comparando-as com robôs e (iii) a capacidade de processamento de pistas visuais, estudando a posição dos objetos e a quantidade de *links* acessados por proximidade de objetos. Eles concluíram que robôs automatizados podem ser diferenciados de humanos com a utilização dos métodos propostos, mas que robôs mais sofisticados poderiam imitar os padrões estudados e se passarem por "humanos" na utilização do método.

Por sua vez, o trabalho de (WANG, 2011) apresentou a criação de um método de simulação baseado especificamente em uma plataforma de *hardware* chamada *Spirent Test Center* para simular o tráfego de rede de um ataque DDoS e de um fenômeno de *Flash Crowd*. Os autores mostraram resultados empíricos, incluindo a simulação de quatro tipos de ataques DDoS, entre eles o ataque UDP *Flood*, o ataque ICMP *Flood*, o ataque TCP SYN *Flood* e o ataque *App-DDoS*. A conclusão do trabalho mostrou que a geração do tráfego de um ataque DDoS e de um *Flash Crowd* através do *hardware* não pode ser comparada a acessos reais de *hosts* legítimos devido a limitações da plataforma utilizada e, com isso, não pôde determinar as características reais dos dois fenômenos analisados.

Já o trabalho de (YU, 2012) estudou profundamente o tamanho e a organização das

botnets atuais e verificou que os fluxos de ataques são geralmente mais semelhantes um ao outro em comparação com os fluxos de fenômenos de *Flash Crowd*. Com base nisto, eles demonstraram um algoritmo de discriminação usando o coeficiente de correlação de fluxo como uma métrica de similaridade entre os fluxos suspeitos. Além disso, eles formularam o problema e apresentaram provas teóricas para a viabilidade do método proposto. Utilizando experimentos extensos, confirmaram a análise teórica e demonstraram a eficácia do método proposto na prática. No final do trabalho, os autores comentaram sobre as "anti-detecções" e na possibilidade de um atacante, conhecendo o método criado, construir uma *botnet* que possa "burlar" o método proposto.

O trabalho de (THAPNGAM, 2012) propôs uma detecção baseada em comportamentos que podem discriminar o tráfego de ataques DDoS do tráfego gerado por usuários reais. Por meio do coeficiente de correlação de Pearson, o método de detecção pode extrair as características que se repetiam na chegada dos pacotes. Eles executaram simulações que foram testadas para verificar a precisão da detecção. Após isto, foram realizados experimentos com vários conjuntos de dados e os resultados demonstraram que o método proposto pode diferenciar o tráfego de um ataque DDoS do tráfego legítimo (*Flash Crowd*), com uma resposta rápida. No final, também foram discutidas outras abordagens para melhorar os métodos propostos, visto que, em alguns cenários, o método apresentado não atingiu os objetivos esperados.

Finalmente, o trabalho de (JEYANTHI, 2013) desenvolveu um mecanismo para a detecção de ataques DDoS visando uma otimização de lucro para os provedores de serviços de computação em nuvem. Para isto, os autores propuseram um algoritmo chamado *Escape-Onsight* (EoS) que visa detectar as características do atacante, analisando as condições do tráfego etapa por etapa e que tem como objetivo proteger o *Data Center* do tráfego malicioso. A análise mostra que a abordagem proposta consegue um bom nível de detecção dos ataques DDoS, trazendo vantagens lucrativas para os provedores de serviços de computação em nuvem que são propensos a esses ataques.

Enfim, podemos verificar que o problema de diferenciação das anomalias ainda não foi resolvido e que pesquisas recentes estão sendo elaboradas. Com o crescimento da Web, métodos para a diferenciação de ataques de DDoS e fenômenos de *Flash Crowd* estão se tornando cada vez mais importantes e estudados, uma vez que as diferenças entre ambos os fenômenos estão cada vez menores. A proposta apresentada nesse trabalho demonstra um método de diferenciação de ataques DDoS Web e *Flash Crowds* que eleva

o nível de detecção das anomalias para a camada de apresentação do *site*-alvo. Nesse caso, o cliente legítimo que acessa o *site* é responsável pela classificação das anomalias através de uma autenticação gráfica que será solicitada sempre que o *site* sofrer grandes quantidades de acessos. Dessa forma, ataques DDoS Web que forem efetuados imitando todas as características dos acessos legítimos serão detectados, assumindo-se que os robôs utilizados nos ataques não possuem inteligência para resolver os testes. Como os ataques DDoS Web são cada vez mais parecidos com acessos legítimos, os trabalhos anteriores que se baseiam nas diferenças entre ambos não conseguem diferenciá-los com precisão, fazendo com que esse trabalho traga uma importante contribuição para a resolução deste problema.

3 METODOLOGIA

Esse capítulo apresenta a metodologia proposta para diferenciar os fenômenos de *Flash Crowds* de ataques DDoS Web através de uma análise mais detalhada dos *sites* da Web que são alvos desses fenômenos. Por exemplo, quando um grande acidente climático ocorre, o número de acessos aos *sites* de notícias normalmente aumenta de forma abrupta, ou mesmo quando uma nova versão de um *software* famoso é liberada, o tráfego trocado pelo servidor de *download* será muito maior do que o normal. Assim, os trabalhos existentes na literatura não fazem a análise dos possíveis motivos lícitos que levaram ao aumento do tráfego da rede. Esta análise está se tornando cada vez mais importante, uma vez que as diferenças entre ambos os fenômenos são cada vez menores, sendo necessário elevar o nível de detecção para uma camada superior (características dos *sites* e usuários), aumentando a qualidade na identificação desses eventos. Portanto, a proposta deste trabalho considera os metadados (MILSTEAD, 1999) extraídos dos *sites*-alvo, um método de autenticação gráfica para os usuários, além das características de tráfego de rede que serão utilizadas como um gatilho para a análise efetuada.

A seguir serão apresentadas a análise integrada, o modelo para o cálculo das probabilidades, o autoaprendizado e o método de autenticação gráfica para a detecção e diferenciação dos fenômenos de *Flash Crowd* e ataques DDoS Web.

3.1 ANÁLISE INTEGRADA

A análise integrada do tráfego de rede e das características dos *sites* atacados visa melhorar a precisão na diferenciação dos fenômenos de *Flash Crowd* e ataques DDoS Web. Pode-se exemplificar essa análise observando o caso de um *site* da Web que foi modificado horas antes de sofrer um alto nível de acessos e, além disso, tem como característica uma alta relevância na Web. Esses dados, aliados à informação de que o número de acessos aumentou consideravelmente, indica que o *site* provavelmente é vítima de um fenômeno de *Flash Crowd*. Um contraexemplo seria um *site* da Web que tenha a mesma quantidade de acessos, porém sem modificações recentes em suas páginas e que possui uma relevância muito baixa. Esse *site* é muito provavelmente vítima de um ataque DDoS Web.

3.1.1 ANÁLISE DAS CARACTERÍSTICAS DOS *SITES* (METADADOS)

Definem-se por metadados os dados sobre os próprios dados, ou seja, informações úteis para identificar, localizar, compreender e gerenciar os dados. Em nosso caso, os metadados estão relacionados às características dos *sites* da Web e que são fundamentais para se melhorar a diferenciação dos fenômenos estudados nesse trabalho. Esses metadados são armazenados em um repositório de dados à medida que há necessidade de coleta dessas informações. Basicamente, duas informações são coletadas dos *sites* da Web: data de modificação e relevância. Essas características foram escolhidas por serem muito importantes na distinção de um fenômeno de *Flash Crowd*, visto que, um *site* da Web que foi modificado recentemente ou que tem um nível de relevância muito alto, tem maior probabilidade de ser vítima de *Flash Crowd*, por exemplo. Assim, a análise dos metadados dos *sites* envolve:

- Data de Modificação do *Site*: se um *site* não sofre atualizações há muito tempo, é normal que o *site* não sofra um aumento de acessos de forma inesperada. Caso esse *site* tenha uma quantidade anormal de acessos, significa que o *site* tem muita probabilidade de ser alvo de um ataque DDoS Web. Porém, se um *site* sofreu modificações recentes e passar por uma explosão de acessos simultâneos após essa modificação, é bem provável que ele seja um possível alvo de *Flash Crowd*, pois os acessos se dão pela "novidade" que aquele *site* apresenta. A vantagem dessa análise pode ser observada na caracterização de ataques DDoS Web, pois *sites* que não sofreram modificações recentes são muito provavelmente alvos desses ataques. A desvantagem dessa métrica pode ser observada na detecção de fenômenos de *Flash Crowd*, pois um *site* que foi modificado recentemente não necessariamente pode ser vítima desse fenômeno.

- Relevância do *Site*: dadas todas as características que um *site* da Web pode apresentar, a relevância do mesmo é sem dúvida uma das mais importantes métricas para classificação dos fenômenos de acesso que o *site* pode sofrer. Relevância é o grau de importância que um determinado *site* da Web possui, extraído de mecanismos de busca na Web (LANGVILLE, 2006). A relevância de um *site* determina a posição que um link de uma página ocupará no resultado de uma busca para um determinado termo (palavra-chave) pesquisado. Assim, quanto maior a relevância de um *site*, melhor será a posição ocupada por ele no resultado da busca. Como o

Google é o mecanismo de busca mais utilizado na Internet, ele foi escolhido para calcular as medidas de relevância do *site*. A relevância que o Google determina é um elemento que seleciona os *sites* que provavelmente irão sofrer fenômenos de *Flash Crowd* e, por isso, *sites* pouco relevantes não irão aparecer no topo das buscas do Google, fazendo com que as pessoas não tenham acesso ao mesmo. Por sua vez, *sites* mais relevantes também podem ser alvo de ataques DDoS por serem mais relevantes e, consequentemente, mais famosos.

3.2 MODELO PARA O CÁLCULO DAS PROBABILIDADES

O modelo proposto nesse trabalho utiliza a estatística bayesiana para interpretar a probabilidade de uma anomalia de rede ser um fenômeno de *Flash Crowd* ou um ataque DDoS Web. O modelo bayesiano (DARWICHE, 2010) permite representar quantitativamente um grau de certeza sobre as incertezas e manipular essas representações segundo as leis da probabilidade clássica. As redes bayesianas possuem várias vantagens para análise de dados. Uma vez que o modelo codifica as dependências entre variáveis, ele manipula facilmente situações onde alguns dados estão incompletos (FUNG, 1995). Além disso, uma rede bayesiana pode ser utilizada para aprender relacionamentos causais e, portanto, pode ganhar conhecimento sobre um domínio de problema e também fazer inferências sobre ele (HECKERMAN, 1995). Para a aquisição de conhecimento, foi utilizada uma aprendizagem por reforço, que será apresentada mais adiante. A aprendizagem por reforço é inspirada na forma como os seres humanos costumam aprender durante a maior parte de sua vida, isto é, através da interação direta com o meio ambiente (SUTTON, 1998) (KAELBLING, 1996). Como os dados iniciais foram definidos de forma empírica, devido à impossibilidade de quantificar de forma exata os parâmetros do modelo, a utilização da aprendizagem por reforço se fez necessária. Nesse caso, o sistema aprenderá as medidas de probabilidade de acordo com os erros e acertos obtidos nas tomadas de decisão. O objetivo do modelo bayesiano proposto nesse trabalho é calcular as probabilidades associadas a dois eventos de rede específicos, com base em dados pré-estabelecidos. A Tabela 3.1 ilustra a probabilidade de ocorrência de *Flash Crowds* e ataque DDoS Web, dadas as informações de modificação e relevância de um determinado *site* da Web. Essas probabilidades foram criadas arbitrariamente e servem de base inicial para a construção do modelo bayesiano.

Para classificar a relevância do *site*, foram utilizados cinco níveis de abstração: relevância

TAB. 3.1: Probabilidades na detecção das anomalias.

Modificação do *Site*	**Relevância do *Site***	***Flash Crowd***	**DDoS**
Muito recente	Muito alta	95%	05%
Muito recente	Alta	85%	15%
Muito recente	Média	80%	20%
Muito recente	Baixa	75%	25%
Muito recente	Muito baixa	65%	35%
Recente	Muito alta	80%	20%
Recente	Alta	70%	30%
Recente	Média	65%	35%
Recente	Baixa	60%	40%
Recente	Muito baixa	50%	50%
Médio	Muito alta	65%	35%
Médio	Alta	55%	45%
Médio	Média	50%	50%
Médio	Baixa	45%	55%
Médio	Muito baixa	35%	65%
Antigo	Muito alta	50%	50%
Antigo	Alta	40%	60%
Antigo	Média	35%	65%
Antigo	Baixa	30%	70%
Antigo	Muito baixa	20%	80%
Muito antigo	Muito alta	35%	65%
Muito antigo	Alta	25%	75%
Muito antigo	Média	20%	80%
Muito antigo	Baixa	15%	85%
Muito antigo	Muito baixa	5%	95%

muito alta, alta, média, baixa e muito baixa. A Tabela 3.2 ilustra os níveis de relevância de um *site* associados ao *PageRank* atribuído pelo sistema do Google (PAGE, 1999). O *PageRank* avalia a importância relativa dos *sites* da Web, proporcionando aos usuários resultados altamente pertinentes com o tipo de busca efetuada no Google.

A data de modificação do *site* também foi classificada em cinco níveis de abstração: muito recente, recente, médio, antigo e muito antigo. A Tabela 3.3 mostra os níveis de modificação do *site* associados ao período de tempo que o *site* sofreu modificações.

Após a união de todos os dados apresentados anteriormente, a rede bayesiana foi empregada para calcular as probabilidades de ocorrência das anomalias de rede estudadas nesse trabalho. A construção do modelo envolveu: (i) a transformação da ocorrência das anomalias em variáveis que pudessem ser representadas no modelo; (ii) a construção

TAB. 3.2: Níveis de relevância do *site*.

Nível de Relevância	**PageRank**
Muito alta	10 a 8
Alta	6 a 7
Média	4 a 5
Baixa	2 a 3
Muito baixa	0 a 1

TAB. 3.3: Níveis de modificação do *site*.

Nível de Modificação	**Tempo de Modificação**
Muito recente	Até um dia
Recente	De um dia até cinco dias
Médio	De cinco dias até duas semanas
Antigo	De duas semanas até um mês
Muito antigo	Acima de um mês

do modelo bayesiano para raciocinar sobre os dados do problema; (iii) o armazenamento do modelo de forma que ele possa ser atualizado continuamente. Todo esse processo de construção do modelo bayesiano é chamado de descoberta de conhecimento. Para viabilizar a construção do modelo foi utilizada a *shell* Netica (NORSYS, 2011), que é um *software* desenvolvido para a criação e manipulação de redes bayesianas e foi projetado para ser simples, confiável e de alto desempenho. A Figura 3.1 ilustra o modelo proposto transformado em uma rede bayesiana. A rede demonstra as probabilidades a priori – não condicionais – associadas às variáveis do problema.

No modelo inicial, temos as características do *site* condicionadas à probabilidade de ocorrência das anomalias de rede. Esse modelo é baseado em uma poli-árvore, ou seja, um grafo direcionado acíclico com a propriedade de, para todo par de nós, ter no máximo um caminho entre os nós. Com isso, podemos calcular as probabilidades dos eventos ocorrerem em tempo linear. A Figura 3.2 demonstra a rede bayesiana de acordo com as probabilidades associadas às variáveis do problema estudado. Quando configuramos o modelo com a relevância e a modificação do *site* para muito alta e recente, respectivamente, tem-se que a probabilidade condicional do fenômeno de *Flash Crowd* acontecer equivale a 80%. Nessas mesmas condições, ataques de DDoS Web têm probabilidade de ocorrência de 20% apenas.

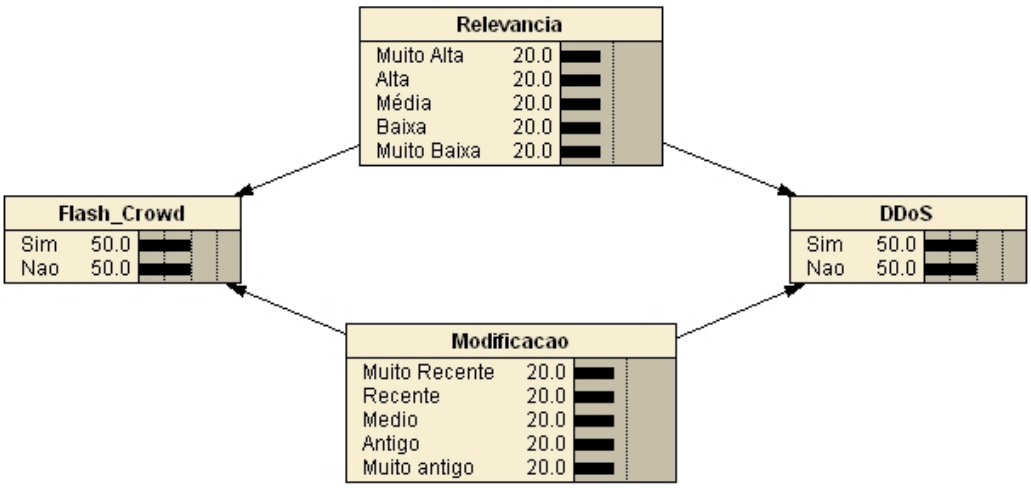

FIG. 3.1: Rede bayesiana inicial.

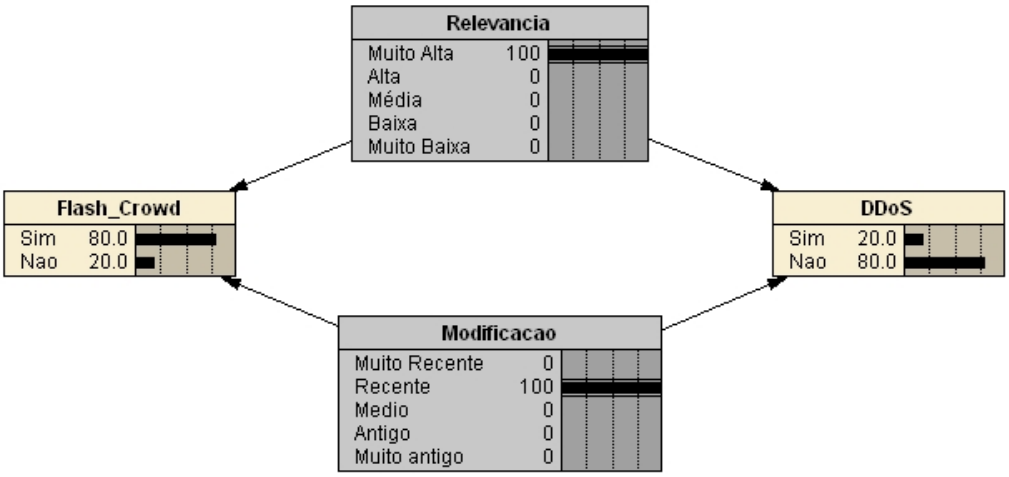

FIG. 3.2: Rede bayesiana condicionada as características do *site*.

3.2.1 ESTUDO DE CASO

Para analisar a viabilidade do modelo apresentado, propõe-se utilizar dados reais, coletados de eventos anteriores que foram vítimas de tais fenômenos, e aplicar o método proposto para verificar quais resultados são obtidos. Um caso bem conhecido na comunidade científica ocorreu por ocasião da Copa do Mundo de 1998. Análises do tráfego de rede foram suficientes para constatar a presença de uma grande quantidade de acessos ao *site* da Copa do Mundo no período de um jogo da semifinal. Nesse momento, a relevância do *site* referente ao jogo era muito alta e sua data de modificação muito recente, carac-

terizando o fenômeno de *Flash Crowd*. Entretanto, no mesmo período, mas em ocasiões diferentes, houve um ataque DDoS ocorrido também em função do evento, porém em condições diferentes - já que o conteúdo do *site* atacado foi um conteúdo estático que não era modificado há bastante tempo. Nesse cenário, tem-se uma relevância alta do *site* com a data de modificação muito antiga. A Figura 3.3 ilustra o modelo proposto sendo utilizado no problema de determinação das anomalias encontradas.

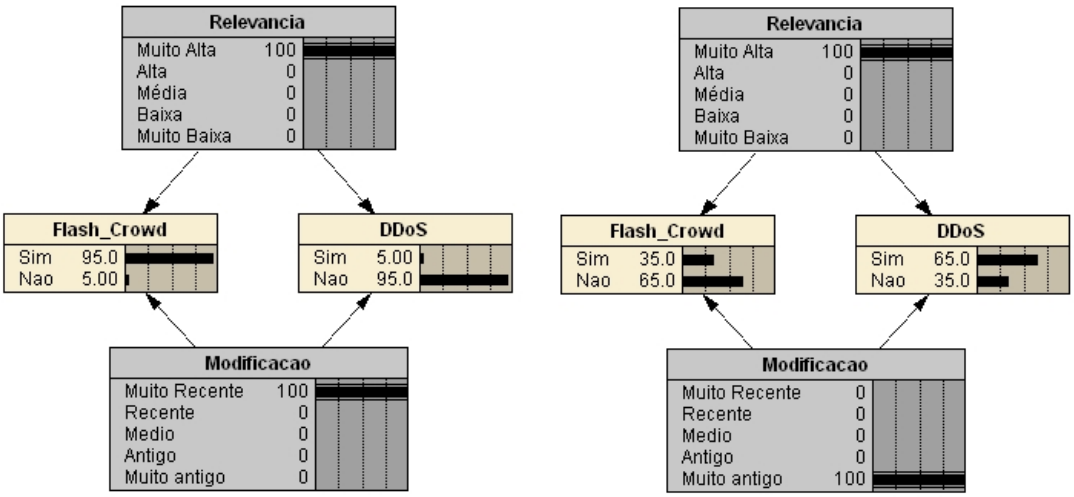

FIG. 3.3: Detecção de *Flash Crowd* e Ataque DDoS respectivamente.

O modelo proposto neste trabalho sugeriu corretamente a maior probabilidade para *Flash Crowd* no primeiro caso e para DDoS no segundo caso. Se este modelo fosse utilizado na detecção das anomalias no advento da Copa do Mundo de 1998, as anomalias de rede poderiam ser classificadas mais precisamente e ações mais eficazes poderiam ter sido tomadas para prevenir a indisponibilidades do *site*, por exemplo. É importante considerar que todo modelo probabilístico é propenso a falhas, mas, com o autoaprendizado customizado para cada *site*, pode-se obter um alto grau de precisão e redução de falso-positivos e falso-negativos. O autoaprendizado será implementado valendo-se da técnica de autenticação gráfica para classificar os acessos maliciosos e adaptar o modelo probabilístico de acordo com a demanda de acessos.

3.2.2 AUTOAPRENDIZADO

A arquitetura descrita neste trabalho utiliza uma técnica de inteligência artificial chamada de aprendizado por reforço (AR), que tem por objetivo modelar a rede bayesiana de forma dinâmica e autônoma. O aprendizado por reforço permite ao sistema adquirir

uma capacidade de conhecimento das probabilidades que não estava disponível a priori. A técnica é baseada na existência de um agente externo ao ambiente, em nosso caso a autenticação gráfica, que avalia a resposta do teste gráfico indicando explicitamente a ação correta a um determinado tipo de acesso. A Figura 3.4 mostra a estrutura utilizada pelo aprendizado por reforço.

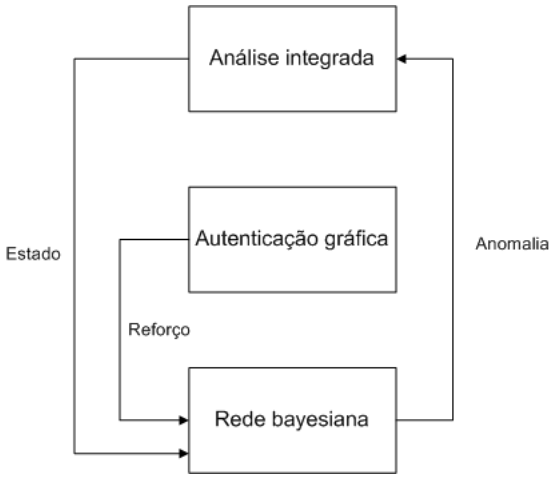

FIG. 3.4: Aprendizado por reforço aplicado ao sistema.

Dois estados marcam o autoaprendizado: treinamento e utilização. Quando em treinamento, o sistema utiliza uma autenticação gráfica para adaptar as probabilidades das anomalias ao modelo bayesiano. Em um tempo pré-determinado, o sistema utilizará essa configuração para aprender quais páginas do *site* são mais propensas a passar por um fenômeno de *Flash Crowd* e quais são mais inclinadas à passarem por um ataque DDoS Web. Nesse momento, o modelo não será utilizado para classificar as páginas e apenas modificará as probabilidades de acordo com as respostas enviadas pelos usuários. Caso o teste seja respondido corretamente, será aplicada uma recompensa na probabilidade do tipo de página requisitada, pois é entendido que o acesso é legítimo. O objetivo da recompensa é aumentar a chance dos acessos àquele endereço serem classificados como um fenômeno de *Flash Crowd*. Se a resposta do teste for incorreta, será aplicada uma punição na probabilidade e os acessos àquela página serão classificados como maliciosos, aumentando a chance do *site* ser vítima de um ataque DDoS Web. A finalidade das métricas de pontuação é adaptar as probabilidades das anomalias ao tipo de *site* acessado, obtendo-se uma classificação mais precisa das mesmas. As métricas de pontuação do autoaprendizado são parametrizadas conforme as características do *site* analisado e poderão ser configuradas de acordo com as necessidades do sistema.

O próximo estado do autoaprendizado é a utilização do sistema. Nessa etapa, o sistema apenas classificará quais os acessos devem ser permitidos e quais são bloqueados. De acordo com as probabilidades aprendidas na fase do treinamento, o sistema diferenciará os acessos de acordo com o tipo de página que os clientes estão requisitando. A Figura 3.5 demonstra o ciclo de treinamento e utilização do autoaprendizado.

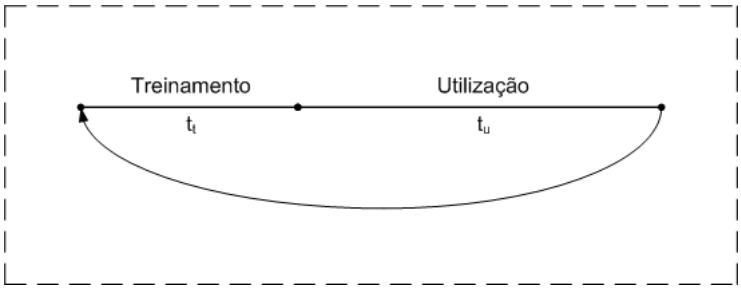

FIG. 3.5: Autoaprendizado: treinamento e utilização.

Os tempos utilizados para o treinamento t_t do autoaprendizado e utilização do sistema t_u são parametrizados de acordo com as características do *site* analisado e podem ser alterados conforme as necessidades do sistema.

3.3 AUTENTICAÇÃO GRÁFICA PARA O AUTOAPRENDIZADO

Esse trabalho propõe um novo método para a detecção de fenômenos de *Flash Crowd* e ataques de DDoS Web. O método consiste num mecanismo de autenticação gráfica (VONAHN, 2003) para detectar os acessos maliciosos feitos por robôs. Para isso, dada algumas condições, a cada novo acesso é requisitada ao usuário uma autenticação por meio de um teste gráfico. Humanos podem resolver facilmente o teste para se autenticarem, enquanto robôs não. Clientes legítimos irão resolver o teste ou recarregar a página algumas vezes a fim de navegarem no *site*. Caso não consigam ou não queiram se autenticar, voltarão ao *site* depois. Já os robôs continuarão enviando novas requisições ao *site* mesmo não conseguindo se autenticar. Dessa maneira, a diferença de comportamento entre os usuários legítimos e os robôs será utilizada para identificar os endereços IPs que pertencem aos robôs e bloquear as suas requisições. Para armazenar os IPs bloqueados, é utilizado um filtro de Bloom (BLOOM, 1970) que possui algoritmos que permitem testar se um endereço IP pertence ao conjunto de dados. Os pedidos de um determinado cliente são descartados se o número dos testes não resolvidos exceder um limite determinado, k. Quando os acessos ao servidor voltarem ao fluxo normal e o servidor operar com a carga

de trabalho normalizada, a autenticação não será mais solicitada. Nesse momento, apenas as requisições cujos IPs estiverem no filtro de Bloom serão descartadas e os usuários legítimos que não resolveram os testes terão acesso ao conteúdo do *site* novamente. O filtro de Bloom será limpo periodicamente pois a maioria dos IPs utilizados em ataques são dinâmicos e clientes legítimos podem adquirir esses IPs após um ataque. A Figura 3.6 mostra o fluxo básico do método proposto até a ativação da autenticação gráfica.

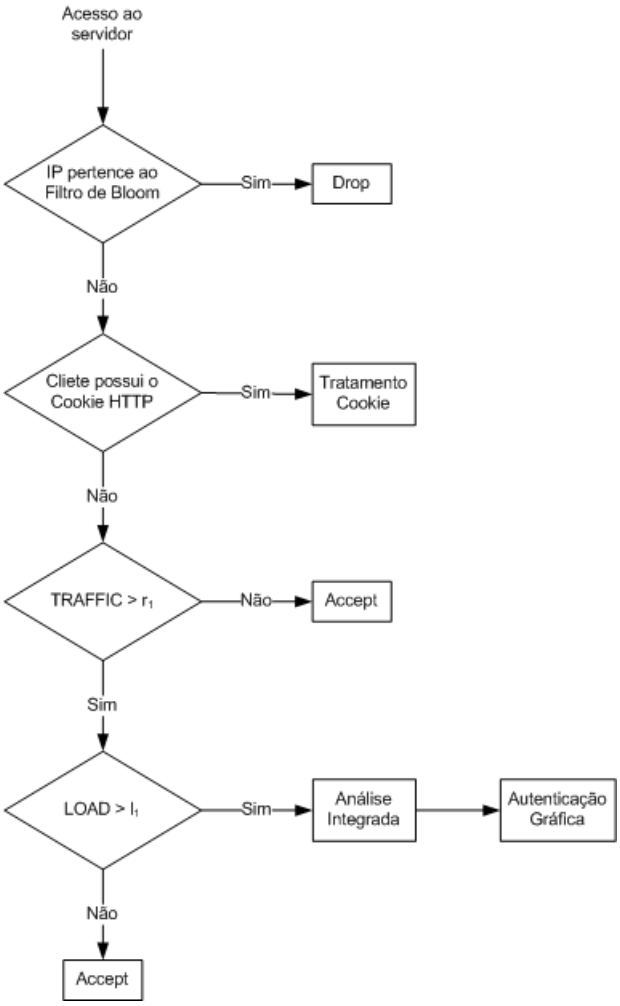

FIG. 3.6: Fluxo básico do método proposto.

Quando uma nova requisição chegar ao servidor, primeiramente será verificado se o IP do cliente está listado no filtro de Bloom. Se o IP do cliente não for reconhecido como um robô, o seu acesso será permitido a menos que o tráfego de rede do servidor seja alto e o servidor esteja operando com uma grande carga de trabalho. Caso a carga do servidor e o tráfego de rede sejam altos, o sistema fará uma análise integrada das características da página solicitada e, se for necessário, solicitará ao cliente uma autenticação através de um

teste gráfico. Se o cliente responder o teste, ele receberá um *cookie* HTTP (KRISTOL, 2001) que permitirá o acesso ao *site* por um período de tempo, sem a necessidade de resolver os testes novamente: requisições oriundas que possuírem o *cookie* HTTP serão admitidas imediatamente, desde que possuam um TOKEN válido e satisfaçam a algumas restrições de tempo que serão tratadas a seguir.

3.3.1 ATIVANDO A AUTENTICAÇÃO

Os acessos ao servidor Web são medidos em função da frequência normal com que as requisições HTTP são processadas pelo servidor. Quando o sistema verifica que os acessos HTTP estão aumentando e passando de um limite normal aceitável, $TRAFFIC > r_1$, ele muda para o modo de operação HIGH_TRAFFIC. Nesse modo, o servidor Web está recebendo mais requisições do que o normal e um fenômeno de *Flash Crowd* ou um ataque DDoS Web pode estar acontecendo. Ao verificar que os acessos estão aumentando, o sistema passará para a próxima fase que analisará a carga de trabalho que o servidor Web está operando. Quando os acessos ao servidor se normalizarem, $TRAFFIC <= r_2$, o servidor voltará a operar em NORMAL_TRAFFIC e as novas requisições serão permitidas desde que não estejam armazenadas no filtro de Bloom. A definição dos valores para r_1 e r_2 deve ser realizada com base no histórico e nos limites aceitáveis de operação do servidor Web e pode variar de acordo com o tipo de *site* analisado. A Figura 3.7 ilustra os modos de tráfego do servidor Web e suas transições.

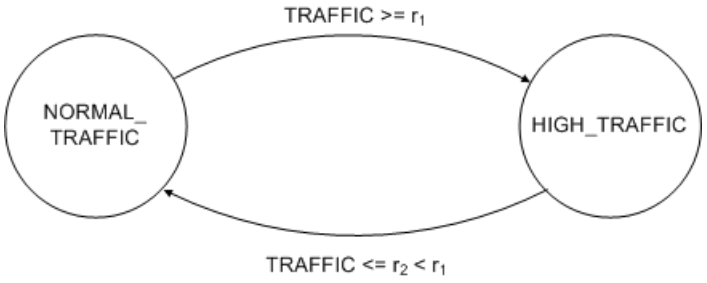

FIG. 3.7: Modos de tráfego HTTP do servidor Web.

O servidor também possui dois modos de utilização: NORMAL_LOAD ou HIGH_LOAD. Quando o servidor Web perceber que seus recursos computacionais passaram de um limite aceitável, l_1, ele muda para o modo HIGH_LOAD. Nesse modo, todas as novas requisições serão processadas utilizando a análise integrada das características dos *sites* e do tráfego de rede do servidor. Caso o sistema esteja em treinamento, isto é, o modelo probabilístico está se adaptando as condições de acesso do *site*, o usuário será imediatamente

direcionado para um teste gráfico que permitirá a sua autenticação no sistema. Quando o usuário responder o teste e se autenticar, o servidor garantirá que os acessos oriundos dele serão permitidos por um período de tempo t_1. Caso o sistema continue operando em HIGH_LOAD por um grande tempo, um gatilho deverá ser disparado solicitando mais recursos computacionais para o servidor Web. Se o sistema estiver em utilização, ou seja, com a capacidade de diferenciar as anomalias, será calculada a probabilidade do acesso ao servidor Web ser um ataque DDoS Web ou um fenômeno de *Flash Crowd*. Para isto, serão utilizados o modelo bayesiano e as tabelas de probabilidade sugeridas anteriormente. Caso a probabilidade seja maior para um ataque DDoS Web, o sistema bloqueará aquele acesso imediatamente para que o mesmo não influencie no desempenho do *site*. Se a probabilidade for maior para um fenômeno de *Flash Crowd*, o acesso será liberado automaticamente e o cliente não precisará responder a nenhum teste. Requisições que iniciarem antes do servidor entrar no modo HIGH_LOAD serão permitidas independente da utilização do servidor, até atingirem o limite de tempo t_2, que é fornecido quando o servidor opera no modo NORMAL_LOAD.

O servidor continuará operando em modo HIGH_LOAD até o nível de carga descer para um intervalo normal e atingir o limite $l_2 < l_1$. A carga de trabalho do servidor é medida em função da utilização de CPU e memória do mesmo, tendo como base a capacidade média dos recursos do servidor. Os valores de l_1 e l_2 podem variar, dependendo da capacidade de processamento e quantidade de memória do servidor. Podemos utilizar, por exemplo, valores de 80% e 60% para l_1 e l_2, respectivamente. Nesse caso, quando o servidor atingir 80% de utilização dos seus recursos, ele entrará em HIGH_LOAD. Após esse momento, ele só voltará a operar em NORMAL_LOAD quando a carga de trabalho cair para 60%. Esses valores são parametrizados de acordo com as características do *site* analisado e podem ser alterados conforme as necessidades do sistema. A Figura 3.8 ilustra os modos de operação do servidor Web e suas transições.

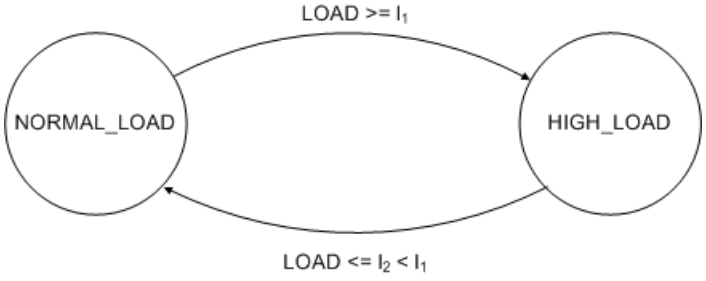

FIG. 3.8: Modos e transições do servidor Web.

3.3.2 TESTES GRÁFICOS

Quando o servidor Web entrar em modo HIGH_LOAD e o sistema estiver em treinamento, a autenticação gráfica é solicitada. Para isso, foi utilizado um teste gráfico de autenticação chamado CAPTCHA (**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part) (VONAHN, 2003). Um sistema de CAPTCHA se consiste de meios automatizados de gerar novos desafios que os robôs são incapazes de resolver, mas a maioria das pessoas podem resolver. Um teste CAPTCHA não confia nunca no atacante que conheça previamente o teste. Por exemplo, um *checkbox* "clique aqui se você é um robô" pode servir para distinguir pessoas e computadores, mas não é um mecanismo CAPTCHA. Para ser um teste CAPTCHA, um sistema deve gerar automaticamente os novos desafios que requerem técnicas da inteligência artificial na resolução. Um tipo comum de CAPTCHA requer que o usuário identifique as letras de uma imagem distorcida, às vezes com a adição de uma sequência obscurecida das letras ou de dígitos que apareçam na tela. Como o teste é administrado por um computador, em contraste ao teste padrão de Turing que é administrado por um ser humano, podemos descrevê-lo como um "teste reverso de Turing" (PUTCHALA, 2011). A Figura 3.9 mostra um exemplo do teste gráfico solicitado pelo servidor.

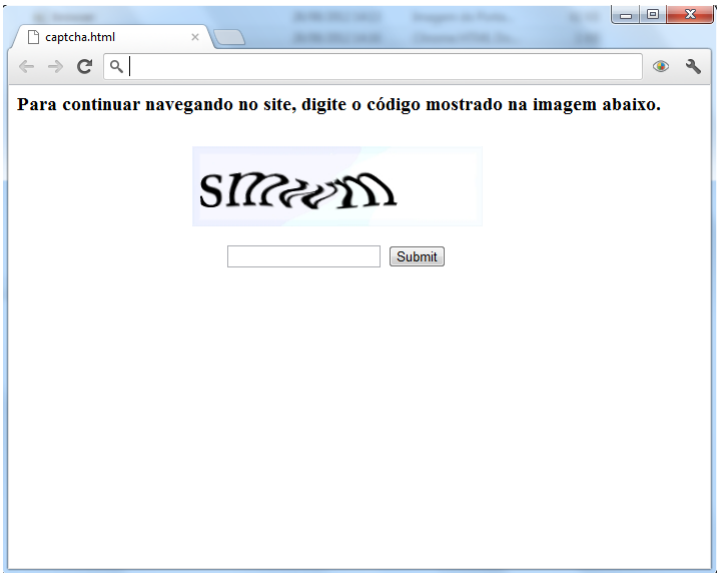

FIG. 3.9: Imagem do teste gráfico.

Inicialmente, o servidor envia para o cliente um formulário contendo o teste gráfico. O cliente então envia uma resposta do teste, correta ou não, com o formulário HTTP GET /validar?ASWER=resposta. Se o formulário não contiver a resposta correta ou

estiver em branco, o servidor responde com um novo formulário de teste gráfico. Caso o cliente responda o teste gráfico corretamente, ele receberá um *cookie* HTTP que conterá um TOKEN de acesso ao servidor Web e o tempo de expiração do teste. A Figura 3.10 mostra o código HTML utilizado no teste gráfico.

```
<html>
<body>
<h3>Para continuar navegando no site, digite o código mostrado na imagem abaixo.</h3>
<form method = "GET" action="/validar">
<img src = "captcha.jpg">
<input type = "password" name = "ANSWER">
<input type="submit" value="Submit" name="submit">
</form>
</body>
</html>
```

FIG. 3.10: HTML do teste gráfico.

Após o envio da resposta correta, o cliente terá seu acesso liberado por uma quantidade de tempo t_1. Caso o tempo t_1 seja atingido e o servidor não tenha entrado em NORMAL_LOAD, a autenticação é solicitada novamente. Isso ocorrerá para todos os clientes, exceto para os clientes que tenham feito requisições enquanto o servidor operava em NORMAL_LOAD. Nesse caso, os clientes recebem um *cookie* HTTP com um tempo t_2. Durante esse tempo, mesmo se o servidor alterar seu modo para HIGH_LOAD, esses clientes continuarão acessando o *site* sem a necessidade de responder o teste gráfico. Quando o tempo t_2 se esgotar, os clientes serão redirecionados automaticamente para o teste de autenticação caso o servidor continue operando em modo HIGH_LOAD. A Figura 3.11 ilustra todas as etapas do sistema e o tratamento do *cookie* HTTP pelo servidor Web, juntamente com as definições de tempo t_1 e t_2.

Cada *cookie* HTTP contém um TOKEN que permitirá fazer apenas oito requisições simultâneas. Isso se deve ao fato de um atacante poder se autenticar e distribuir o *cookie* aos milhares de robôs. Com essa restrição, esse cenário é impossibilitado. Além disso, a maioria dos navegadores Web não abrem mais do que oito conexões simultâneas para cada servidor e por isto não é possível utilizar um número menor do que oito (JAMJOOM, 2003).

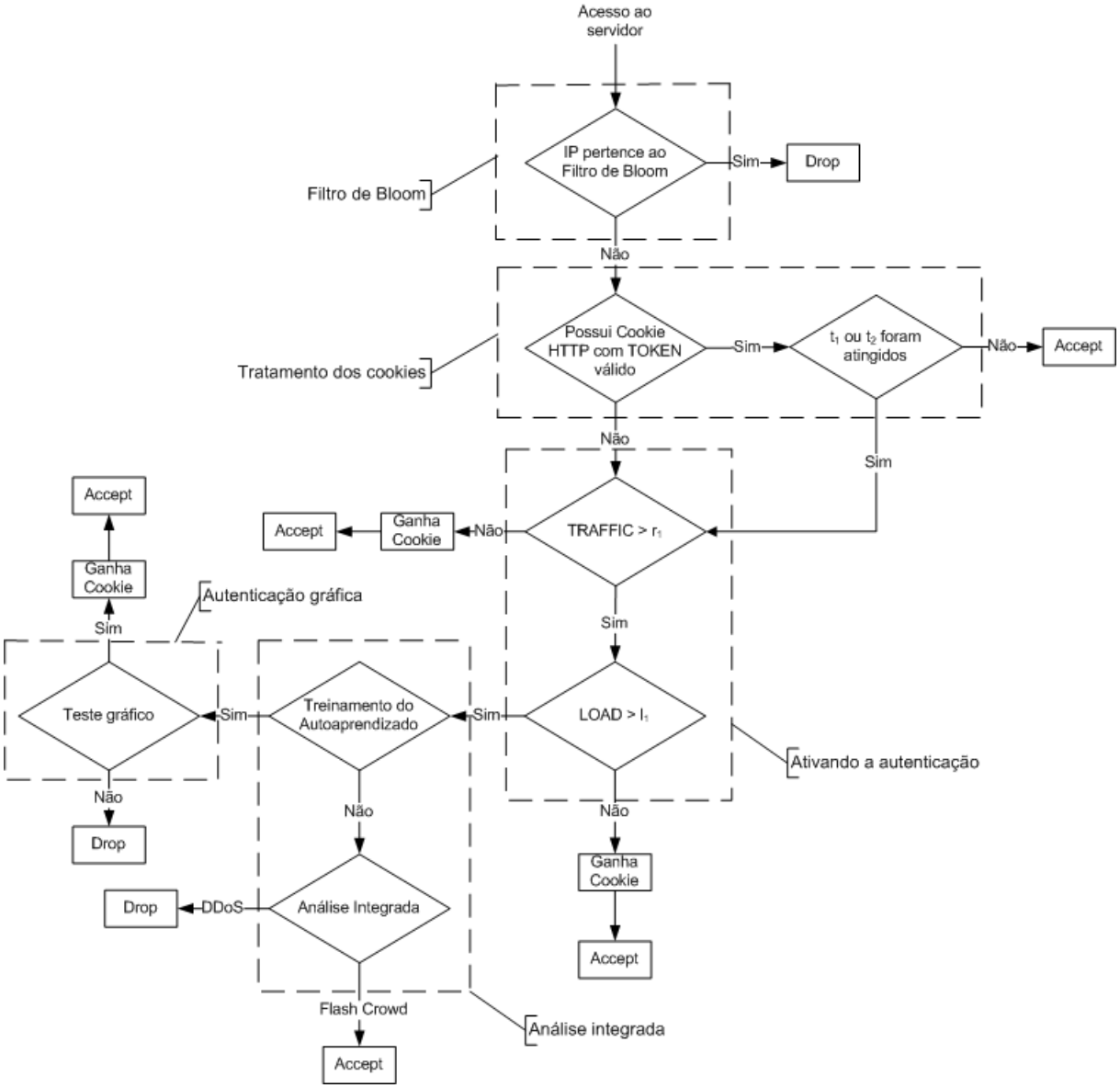

FIG. 3.11: Detalhamento das etapas do sistema.

4 IMPLEMENTAÇÃO

A arquitetura utilizada para a implementação deste trabalho é baseada em camadas e é apresentada na Figura 4.1. Inspirada na arquitetura de um agente de *software* (RUSSELL, 1995), a camada inferior, chamada de Camada de Sensoriamento, atua no sensoriamento do ambiente, fornecendo informações importantes para a camada intermediária, conhecida como Camada de Processamento. Esta, por sua vez, processa as informações coletadas e é responsável pelo gerenciamento das ações a serem tomadas. Finalmente, a camada superior, chamada Camada de Serviço, responsável pela execução dos serviços fornecidos pela arquitetura, atua e modifica o ambiente externo, de acordo com as ações definidas pela Camada de Processamento. Cada uma dessas camadas é descritas em detalhes a seguir.

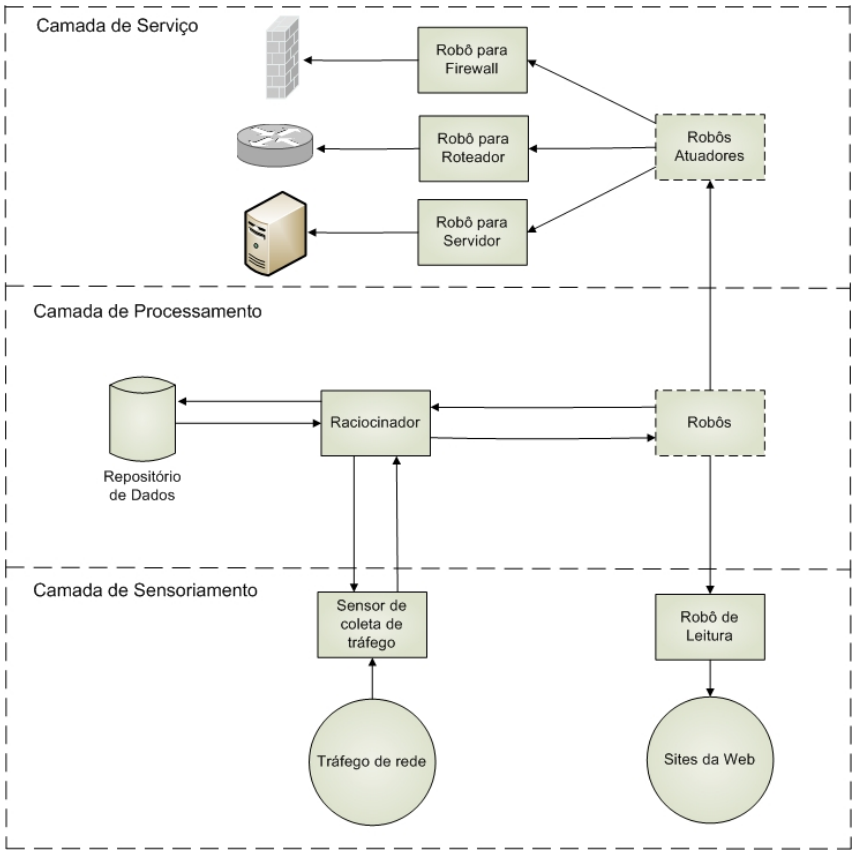

FIG. 4.1: Arquitetura conceitual.

- Camada de Sensoriamento: essa camada está no nível mais baixo da arquitetura e é

responsável pelo monitoramento do tráfego de rede, do servidor Web e do conteúdo dos *sites*-alvo. Na Camada de Sensoriamento estão contidos os sensores de coleta de dados e os robôs de leitura das informações dos *sites* da Web. Os sensores são responsáveis pela coleta de todo o tráfego de rede, pelo nível de carga do servidor Web e pela transmissão dos dados coletados para a camada de serviço, enquanto os robôs são responsáveis pela leitura dos *sites* da Web que podem ser possíveis vítimas de um fenômeno de *Flash Crowd* ou ataques DDoS Web. Os sensores são componentes passivos de coleta de dados e os robôs de leitura são componentes ativos, que coletam os metadados dos *sites* de acordo com a requisição da camada de processamento.

- Camada de Processamento: essa camada é responsável pelo processamento e tratamento dos dados, além da tomada de decisão caso seja detectado um fenômeno de *Flash Crowd* ou um ataque DDoS Web. Na camada de processamento, encontra-se o raciocinador e um repositório para armazenamento dos dados coletados. O raciocinador é o centro da arquitetura e é responsável pela análise das informações coletadas pela camada de sensoriamento e pela tomada de decisão, caso o *site* seja vítima de uma anomalia. Quando o raciocinador analisa os dados oriundos dos sensores e encontra uma possível anomalia, ele ativa os robôs de leitura para uma análise mais precisa das informações do *site* atacado. Essas informações são comparadas com os dados contidos no repositório de dados e utilizadas para ter-se um maior nível de precisão na detecção e diferenciação das anomalias.

- Camada de Serviço: essa camada está no nível mais alto da arquitetura e é responsável pela comunicação do sistema com os ativos da rede local. Quando um fenômeno de *Flash Crowd* ou um ataque DDoS Web é detectado pelo raciocinador, os robôs atuadores são acionados para operarem em um determinado ativo ou grupo de ativos de rede de acordo com as características do fenômeno. Os robôs atuadores podem atuar em firewalls, roteadores e servidores além de enviar alertas para os administradores do *site* informando quando um problema for encontrado.

A análise do tráfego de rede é uma métrica fundamental para o reconhecimento das anomalias estudadas nesse trabalho. Através da coleta e análise do tráfego de rede serão explorados os padrões de tráfego que os atacantes podem utilizar, a fim de criar mecanismos que possam reconhecer as anomalias. Em um primeiro momento, será coletado todo

o tráfego direcionado para o *site* da Web monitorado. Após os pacotes serem capturados pela rede, eles são decodificados e colocados no repositório de dados, organizando, filtrando e decodificando o fluxo de dados. Quando os acessos ao servidor Web aumentarem consideravelmente, o sistema sinalizará à Camada de Processamento que o servidor está operando em alto fluxo, funcionando como um gatilho para a ativação do sistema.

Além da detecção das anomalias estudadas nesse trabalho, a arquitetura proposta tem por objetivo atuar na infraestrutura do *site*-alvo de forma a emitir alertas quando as anomalias forem encontradas e, de forma reativa, executar ações necessárias para minimizar os efeitos gerados pelo fenômeno caracterizado. Através de agentes instalados nos ativos de rede, podem-se criar eventos que, mapeados às anomalias de rede, têm uma ação reativa no dispositivo monitorado. O dispositivo é, de forma automática, induzido por meio de configurações e parâmetros, a reagir ao fenômeno detectado. Pode-se, por exemplo, inibir os acessos oriundos de um ataque DDoS Web através da criação de rotas alternativas àquele ataque ou mesmo alterar parâmetros de configuração do servidor Web para que ele tenha mais disponibilidade quando o *site* sofrer um fenômeno de *Flash Crowd*. Essas ações deverão ser previamente configuradas no sistema e serão aplicadas de acordo com a detecção das anomalias de rede. Além disso, alertas poderão ser enviados aos administradores locais para que os mesmos possam atuar na resolução do problema encontrado. Todas essas medidas visam, além da detecção das anomalias de rede, a prevenção de possíveis danos que um *site* da Web possa sofrer quando os fenômenos estudados nesse trabalho ocorrerem.

4.1 FERRAMENTAS UTILIZADAS

Para viabilizar a implementação da arquitetura proposta, foram utilizadas algumas ferramentas bastante conhecidas que serão brevemente apresentadas a seguir. A Figura 4.2 ilustra a arquitetura de implementação proposta.

Para os sensores de coleta de dados será utilizando o sistema de detecção de intrusão Snort (BEALE, 2004). A escolha dessa ferramenta deve-se ao fato dela permitir a captura de todo o tráfego de rede de um determinado *site* e analisar o conteúdo de todos os pacotes que por ele passam em tempo real. Além disso, o Snort tem o seu núcleo flexível a assinaturas de ataque, isto é, ele possui um banco de dados de ataques constantemente atualizado que pode ser adicionado ou atualizado de acordo com as necessidades do sistema.

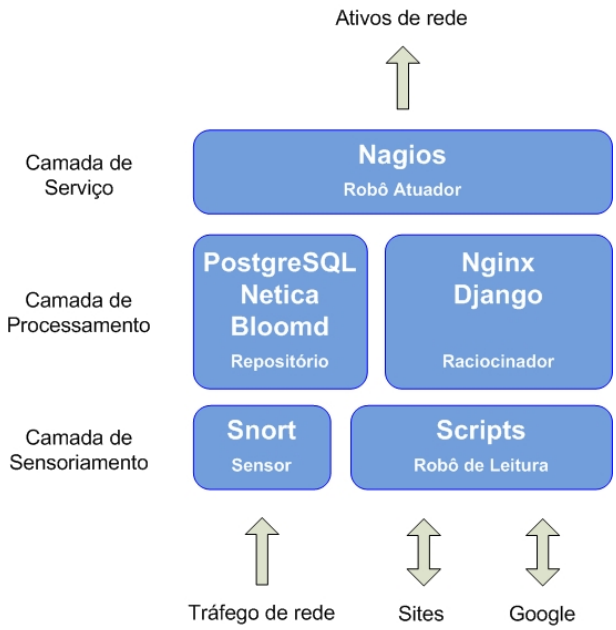

FIG. 4.2: Arquitetura de implementação.

Para os robôs de leitura das informações dos *sites* da Web, além dos comandos do próprio Sistema Operacional utilizados para coletar as informações referentes à data de modificação do *site*, o Google foi utilizado para prover as informações da relevância do *site* da Web através de uma característica chamada *ranking* da página (*PageRank*) (HAVELIWALA, 1999). O *PageRank* avalia a importância relativa dos *sites* da Web, proporcionando aos usuários resultados altamente pertinentes.

O raciocinador foi implementado utilizando como base um servidor de *proxy* reverso. Dentre alguns servidores que possuem essa função, foi escolhido o Nginx (REESE, 2008) para a solução proposta. O Nginx é um servidor Web e de *proxy* reverso criado para suportar grandes picos de acessos mantendo um bom desempenho. Ele foi desenvolvido usando um modelo orientado a eventos assíncronos que provê resultados de desempenho mais previsíveis sob grande demanda, em contraste ao modelo adotado por outros servidores, que por padrão utilizam *threads* ou processos para responder às requisições. Além disso, o Nginx é altamente escalável, uma vantagem essencial ao tipo de sistema proposto neste trabalho.

Além do Nginx como *proxy* reverso, também utilizamos uma aplicação em Python (LUTZ, 2003) para o raciocinador disponibilizada através do *framework* de desenvolvimento Web Django (HOLOVATY, 2007). Python é uma linguagem de programação que tem por objetivo ser de propósito geral. Em outras palavras, pode-se criar com esta

linguagem desde *scripts* para automatizar tarefas até grandes sistemas Web. Devido a facilidade de aprendizado e uma ampla comunidade em torno do projeto, escolheu-se essa linguagem para implementar o sistema responsável pela diferenciação das anomalias estudadas neste trabalho. Por sua vez, o Django foi escolhido por ser um *framework* de desenvolvimento rápido para Web, escrito em Python, que utiliza o padrão MTV (*Model - Template - View*). Ele foi escolhido devido a sua ampla adoção, alto desempenho e facilidade de implementação.

Para o repositório de dados foram utilizadas três tecnologias distintas: o sistema gerenciador de banco de dados (SGBD) PostgreSQL (MATTHEW, 2005), o serviço Bloomd para os filtros de Bloom (PENG, 2003) e a *shell* Netica (NORSYS, 2011) para o modelo bayesiano. O PostgreSQL foi escolhido por ele ser um SGBD objeto-relacional de código aberto, robusto e confiável, além de ser flexível e rico em recursos. Além disso, ele é considerado objeto-relacional por implementar, além das características de um SGBD relacional, algumas características de orientação a objetos, como herança e tipos personalizados. O modelo objeto-relacional foi escolhido por ele ser otimizado para trabalhar com um grande volume de dados, além de viabilizar agilidade e facilidade no acesso e inserção das informações.

O filtro de Bloom foi implementado utilizando o servidor Bloomd. Este serviço é escrito em Python e tem por objetivo armazenar os filtros de Bloom responsáveis pelo armazenamento dos IPs maliciosos. Além de implementar todos os algoritmos necessários a utilização dos filtros de Bloom, o servidor Bloomd utiliza o Memcached (FITZPATRICK, 2004) que armazena os dados na memória do servidor, aumentando o desempenho na leitura e escrita dos mesmos. O Memcached é um sistema distribuído de alto desempenho para armazenamento de objetos na memória, construído para aumentar a velocidade de *sites* dinâmicos, diminuindo a carga de trabalho do servidor.

O modelo bayesiano foi implementado utilizando a *shell* para sistemas especialistas probabilísticos Netica. As redes bayesianas (POURRET, 2008) são redes de conhecimento que permitem calcular a probabilidade de um evento ocorrer condicionado à ocorrência de outro. A vantagem desse modelo é a facilidade de se extrair conhecimento de forma automática a partir de uma base de dados hipotética, contendo as informações coletadas dos *sites* da Web. A *shell* Netica utiliza a tecnologia de redes bayesianas para realizar inferência usando algoritmos rápidos e modernos.

Finalmente, para os robôs atuadores foi escolhida a ferramenta Nagios. A escolha se

deu em função do Nagios ser um serviço que, além de ser um excelente monitor de falhas, elaboração de relatórios e alertas de rede, foi projetado para atuar nos mais diversos ativos de rede de forma a executar procedimentos automáticos de acordo com os problemas encontrados na infraestrutura. O Nagios compete bem contra a maioria das aplicações comerciais, e no parecer de (SCHUBERT, 2008), na maioria dos casos, o Nagios tem um custo menor com um maior nível de eficácia do que muitas aplicações comerciais com a mesma finalidade.

4.2 DESENVOLVIMENTO DO SISTEMA

A primeira fase de implementação do sistema foi construída com a utilização de um *proxy* reverso. Um *proxy* reverso é um servidor HTTP que é colocado na frente de um servidor Web com o objetivo de manipular as requisições à ele transmitidas. Todas as conexões que chegam no *proxy* reverso são endereçadas para um dos servidores Web através de um roteamento feito pelo próprio servidor *proxy*, que pode tratá-las ou encaminhá-las para um servidor Web. Um *proxy* reverso repassa o tráfego de rede recebido para um ou mais servidores, tornando-se a única interface para as requisições externas destinadas a um determinado serviço Web. O nome *proxy* reverso foi escolhido pois ele atua exatamente como oposto de um *proxy* convencional, que age como um despachante para o tráfego de saída de uma rede para a Internet, representando as requisições dos clientes internos para os servidores externos, geralmente localizados na Internet. A Figura 4.3 ilustra o funcionamento de um *proxy* reverso.

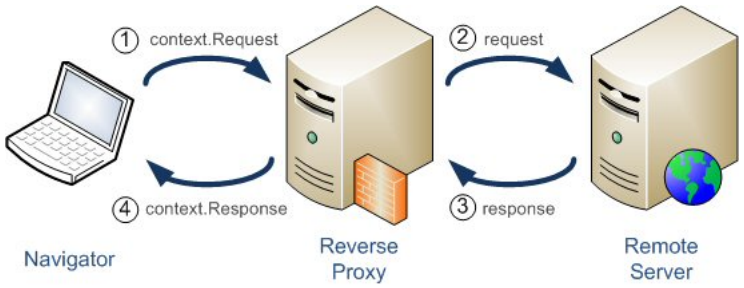

FIG. 4.3: Funcionamento de um *proxy* reverso.

Além do *proxy* reverso, uma aplicação Web foi desenvolvida utilizando o *framework* Django. Essa aplicação foi chamada de StopBots e será o centro da arquitetura do sistema. O StopBots será responsável pela autenticação gráfica e por toda a comunicação dos componentes do sistema, além de fazer a integração entre as ferramentas *open source*

utilizadas na arquitetura. Os filtros de Bloom foram construídos para armazenar os IPs classificados como maliciosos e bloqueá-los caso o sistema esteja com uma carga de trabalho alta. Também foram desenvolvidos *scripts* para monitorar o tráfego de rede e a carga de trabalho do servidor Web, sendo ativados pelo StopBots quando houver necessidade. Além disso, foi criado um algoritmo para implementar a técnica de autoaprendizado por reforço que completará a solução proposta e permitirá um melhor desempenho.

Os robôs atuadores são responsáveis por monitorar a curva de demanda dos servidores Web em tempo real. É possível estabelecer condições para que o raciocinador solicite a autenticação gráfica quando a carga de trabalho dos servidores Web ultrapassar o limite de 80%, por exemplo. Também é possível estabelecer uma condição para remover a autenticação gráfica quando a utilização de recursos dos servidores Web ficar abaixo do limite indicado.

O raciocinador possui a inteligência do sistema e tem a tarefa de controlar os IPs que serão armazenados no filtro de Bloom. Dessa maneira, o servidor Web não tem o seu desempenho comprometido e não precisa processar as requisições de teste gráfico, que são feitas no servidor de *proxy* reverso. Os *cookies* são lidos pelos sensores de coleta de dados e processados pelo StopBots (raciocinador). Caso a requisição atenda as condições de acesso, é permitido o acesso ao *site* da Web diretamente dos servidores Web. Como os servidores não têm seu desempenho comprometido com o controle da autenticação, a utilização média dos mesmos tende a diminuir conforme o ataque DDoS Web for sendo bloqueado. Com isso, os servidores Web não precisam ter sua implementação modificada, facilitando a implementação da técnica proposta e sua adoção. A Figura 4.4 ilustra o modelo proposto com as fases de integração dos processos de implementação.

Os números com as cores em preto mostram o funcionamento do sistema quando o tráfego de rede está normal. Na primeira fase (1) o sistema analisa o tráfego de rede que passa pelo *site* monitorado. Com o fluxo normal (2), o Snort passa a requisição para o StopBots que, por sua vez, verifica se o IP do cliente está no filtro de Bloom (3). Se o IP não estiver no filtro, ele armazena as informações da requisição no PostegreSQL (4) e passa a requisição para o servidor Web (5), fazendo com que o cliente acesse o *site* normalmente.

Os números com as cores em vermelho mostram o funcionamento do sistema quando o tráfego de rede está acima do normal. Na primeira fase (1) o sistema analisa todo o tráfego de rede que passa pelo *site* monitorado. Se o Snort detectar um aumento nos

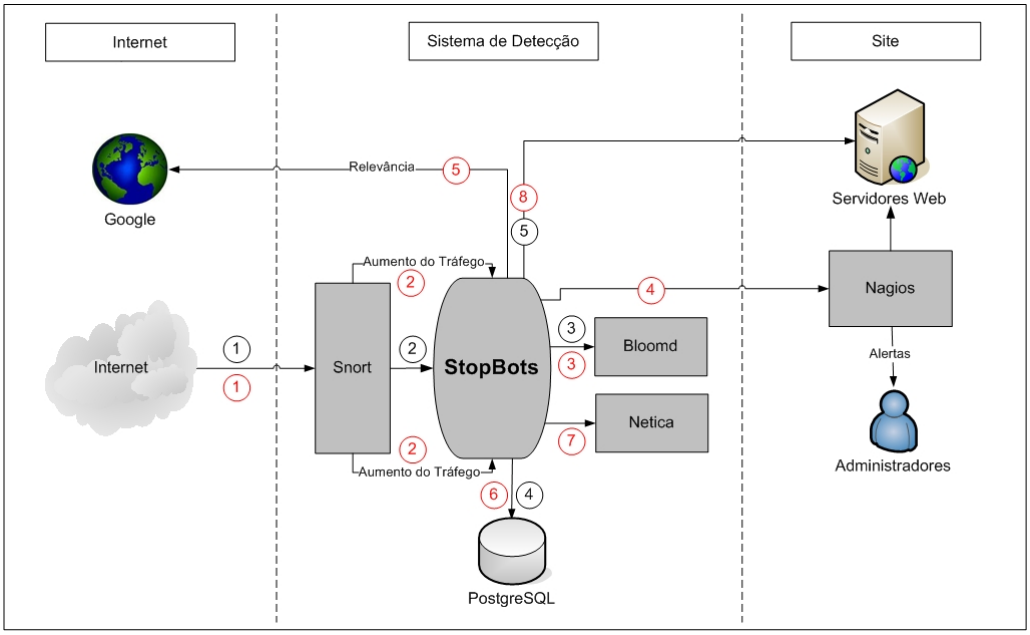

FIG. 4.4: Arquitetura de implementação com o StopBots.

acessos àquele determinado *site* (2), ele envia uma mensagem para o StopBots informando o problema. O StopBots, por sua vez, verifica se o IP do cliente está no filtro de Bloom (3). Caso o IP esteja no filtro de Bloom, o StopBots bloqueia o acesso ao servidor. Se o IP não estiver no filtro, ele solicita ao Nagios informações sobre a carga de trabalho do servidor Web (4). Caso o servidor esteja com a carga de trabalho alta, o StopBots verifica o *PageRank* da página no Google (5) e armazena as informações da requisição no PostgreSQL (6). Após isto, o StopBots verifica se o sistema está na fase de treinamento do autoaprendizado. Caso sim, o StopBots solicita a autenticação gráfica ao usuário e executa as métricas de pontuação do autoaprendizado no modelo probabilístico. Caso não, o StopBots analisa a rede bayesiana (7) e bloqueia ou libera (8) a requisição ao servidor Web de acordo com a classificação da anomalia.

5 EXPERIMENTOS

Para validar a técnica proposta neste trabalho, a arquitetura sugerida foi implementada parcialmente em um ambiente real com o propósito de utilizá-la como possíveis fontes de fenômenos de *Flash Crowds* que eventualmente forem obtidos em determinados momentos de utilização do *site*. Essa escolha se deu em função da dificuldade de simulação de fenômenos de *Flash Crowds* em ambientes de laboratório. A seção 5.1 apresentará a aplicação realizada no sistema para implementá-lo em um *site* de comércio eletrônico varejista na Web. A seção 5.2 demonstrará algumas características do *site* no período analisado, bem como os momentos que o *site* passou por grandes quantidades de acesso. Esses períodos de grandes acessos serão analisados posteriormente para validar a viabilidade da solução. A seção 5.3 apresentará a metodologia empregada para a utilização da técnica e para a coleta dos dados. Nessa parte, serão abordadas as fases de implementação e testes do sistema, bem como as possíveis limitações da análise sugerida. Por último, na seção 5.4, serão apresentados os resultados obtidos e as principais vantagens e desvantagens de utilização da técnica de diferenciação das anomalias estudadas neste trabalho.

5.1 APLICAÇÃO DO SISTEMA EM UM *SITE* DE *E-COMMERCE*

Com o objetivo de viabilizar a utilização da metodologia apresentada em um ambiente real, foi efetuada a aplicação do sistema em um *site* real de *e-commerce* varejista que possui um grande número de acessos diários. Para isto, foi realizada uma análise detalhada da usabilidade do *site*, bem como dos perfis dos usuários que o acessavam e do impacto que a implementação do sistema teria sobre as vendas do *site*. Além da usabilidade propriamente dita, todos os componentes de infraestrutura do *site* foram analisados previamente para que se fosse possível estabelecer valores aos diversos parâmetros necessários ao sistema, como os limites para o servidor entrar em carga de trabalho alta, por exemplo. A Tabela 5.1 ilustra os parâmetros utilizados e os valores atribuídos às variáveis do sistema.

O primeiro parâmetro atribuído foi a quantidade de tentativas que o usuário tem para resolver o teste gráfico. Como o *site* tem o foco na venda de produtos mais populares e baratos, o perfil de usuários que acessam e compram no *site* são de pessoas com pouca

TAB. 5.1: Parametrização do sistema.

Variável	**Valor**	**Definição**
k	5	Número de tentativas para resolver o teste gráfico.
t_1	30 minutos	Tempo de liberação após responder o teste gráfico.
t_2	10 minutos	Tempo de liberação após realizar o primeiro acesso.
r_1	50%	Limite para entrar em HIGH_TRAFFIC.
r_2	20%	Limite para entrar em NORMAL_TRAFFIC.
l_1	70%	Limite para entrar em HIGH_LOAD.
l_2	40%	Limite para entrar em NORMAL_LOAD.
t_t	10 minutos	Tempo de treinamento do autoaprendizado.
t_u	10 minutos	Tempo de utilização do autoaprendizado.

experiência no uso da Internet. Dessa maneira, foi utilizado um valor de cinco de tentativas na resolução do teste gráfico. Isso foi necessário para garantir que pessoas com mais dificuldades pudessem resolver o teste e não fossem classificadas como robôs. Além disso, a autenticação gráfica foi vinculada a uma promoção para motivar o usuário à responder o teste e continuar navegando no *site*. Como o teste gráfico é um tanto quanto intrusivo, essa abordagem foi escolhida para não haver um abandono em massa do *site* quando a autenticação for solicitada e minimizar o impacto da adoção da técnica proposta. A Figura 5.1 demonstra o teste gráfico vinculado a um desconto promocional no *site*.

FIG. 5.1: Teste gráfico vinculado a um desconto no *site*.

Após resolver o teste gráfico, o usuário ganha um tempo $t_1 = 30$ minutos para per-

manecer no *site* sem precisar se autenticar novamente. Esse tempo é determinado para que a autenticação gráfica seja solicitada poucas vezes, visto que ela pode incomodar o usuário e fazer com que o mesmo abandone a navegação do *site*. Também com este propósito, foi criado o tempo $t_2 = 10$ minutos, permitindo que o usuário que já se encontra navegando no *site* não seja incomodado, mesmo que o teste gráfico esteja habilitado. Esse período foi calculado utilizando a média de tempo que os usuários ficam navegando no *site*, que é de aproximadamente 10 minutos.

Os próximos parâmetros determinados foram os limites para o servidor entrar em HIGH_TRAFFIC e voltar para modo NORMAL_TRAFFIC. Analisando os registros de acessos do servidor por meio de *scripts* automatizados, chegou-se a conclusão de que a média normal de acessos ao *site* era determinada em função dos horários que os acessos eram realizados. Dessa maneira, chegou-se a conclusão de que o servidor analisado passa por picos de acesso quando os acessos são 50% maiores do que a média normal do servidor, $r_1 = 50\%$. O limite para o servidor voltar a operar em NORMAL_LOAD, r_2, foi atribuído levando-se em conta que a média de acessos ao mesmo possui 20% de variação nos momentos em que o *site* está com o fluxo de acessos normalizado.

Para a carga de trabalho do servidor, utilizou-se o *Load Average* (GUNTHER, 2004) do Sistema Operacional Linux, que indica o uso dos principais recursos do servidor. Em resumo, o *Load Average* é a média da soma do número de processos aguardando na fila para rodar mais o número atual de processos que estão sendo executados nos últimos minutos. Essa informação tem total relação com a utilização de CPU, memória RAM, I/O de rede, disco ou qualquer outro recurso que o sistema operacional esteja utilizando. Foi verificado que, historicamente, os servidores responsáveis em hospedar o *site* ficam indisponíveis quando passam de 90% de carga de trabalho. Sendo assim, definiu-se que os valores aceitáveis para as variáveis são $l_1 = 70\%$ e $l_2 = 40\%$.

Para o autoaprendizado, foram utilizadas etapas de 10 minutos de treinamento (t_t) e de 10 minutos de utilização (t_u). Isso significa que, quando o servidor entrar em modo HIGH_TRAFFIC, o sistema solicitará a autenticação gráfica durante 10 minutos e, logo após isto, utilizará as classificações aprendidas no treinamento por uma etapa de 10 minutos. Este ciclo será finalizado quando o servidor voltar a operar em modo NORMAL_LOAD, porém as probabilidades serão mantidas até o próximo ciclo de autoaprendizado.

Outra consideração importante é quanto ao bloqueio do acesso quando o mesmo for

necessário. Para o *site* de *e-commerce*, definiu-se que o bloqueio seria vinculado a uma mensagem de "volte mais tarde". Isso faz com que os usuários que não responderam os testes ou tinham seus IPs listados no filtro de Bloom, recebam a informação de que o *site* encontra-se indisponível por alguns minutos mas que posteriormente poderão voltar e continuar comprando, vinculados a um desconto. A vantagem dessa abordagem é que para os robôs essa mensagem será inútil, visto que eles não tem inteligência para entendê-la. Já para os usuários legítimos, o bloqueio momentâneo se torna apenas uma forma de deslocar os acessos bloqueados para um período posterior, tendo o mínimo de influência na conversão de vendas do *site*.

5.2 CARACTERÍSTICAS DO *SITE* DE *E-COMMERCE* NA NUVEM

Essa seção apresenta um estudo da caracterização da carga de trabalho de um *site* de *e-commerce* que utiliza sua infraestrutura na nuvem da Amazon (MURTY, 2008). As métricas desse *site* foram coletadas num período de 90 dias e durante esse período o *site* recebeu mais de 31 milhões de requisições e em alguns momentos passou por grandes picos de acessos.

Sites de *e-commerce* possuem um grande potencial de acessos na Web. Segundo informações da e-bit[2], *sites* que trabalham com comércio eletrônico possuem aproximadamente 32 milhões de clientes e a previsão de crescimento em 2013 é de 25% (e-b). Isso faz com que as informações de acesso a esse serviço sejam uma grande fonte para a caracterização de cargas de trabalho Web.

Esse trabalho apresenta uma caracterização de um grande *site* de *e-commerce* varejista brasileiro. Ao comparar este estudo com resultados de estudos anteriores de caracterização da Web, podemos determinar como as cargas de trabalho Web estão evoluindo à medida que a Web cresce em popularidade e utiliza novas tecnologias.

Algumas das mais significantes características observadas na carga de trabalho Web do *e-commerce* são:

- Clientes HTTP/1.1 são prevalentes, responsáveis por 99,98% de todas as requisições. A implantação de clientes HTTP/1.1 compatíveis foi essencial para o a plena utilização das funcionalidades HTTP/1.1.

[2]Presente no mercado brasileiro desde janeiro de 2000, a e-bit é referência no fornecimento de informações sobre *e-commerce* nacional.

- O método HTTP mais utilizado foi do tipo GET, com 94,41% de todas as requisições.
- As transferências que obtiveram o código de sucesso (200) em sua resposta chegaram a 78,04%, mas também houve muitos erros, sendo em sua maioria o código 404 (*File Not Found*) com 16,15% de todas as requisições.
- De todas as requisições feitas ao *site* durante o período de análise, 66,93% das requisições tiveram sua origem do próprio *site*. Isso quer dizer que mais da metade das requisições foram oriundas da navegação no próprio *site*.
- Das aproximadamente 31 milhões de requisições ao *site*, tivemos 1,39 milhões de endereços IP únicos, ou seja, cada cliente navegou em média por 24 páginas no *site*.

5.2.1 CONJUNTO DE DADOS

O conjunto de dados utilizado neste estudo de caracterização é constituído pelos registros de acessos coletados em cada um dos servidores utilizados no *site* de *e-commerce*. Os registros de acesso de cada servidor foram arquivados em uma base diária. Para este estudo, os *logs* de acesso analisados foram coletados entre 01 de dezembro de 2012 e 28 de fevereiro de 2013. Cada registro de acesso está no formato de *log* comum (w3c). Para cada solicitação recebida pelo servidor Web, a seguinte informação é armazenada:

```
remotehost date method request version code bytes referer "useragent"
```

Esses campos são definidos a seguir:
remotehost: o IP de origem do cliente que faz a requisição.
date: a data e hora da requisição.
method: o método da requisição.
request: o endereço completo da requisição.
version: a versão do protocolo HTTP.
code: o código de resposta HTTP retornado pelo servidor.
bytes: o tamanho do conteúdo do documento transferido.
referer: o endereço anterior de onde a requisição foi originada.
useragent: o nome da aplicação, versão, sistema operacional e outras características do

computador que originou a requisição.

Um exemplo de um registro de acesso é:

```
187.115.4.4 01/Dec/2012:00:00:00 GET http://www.site.com.br/busca/TV HTTP/1.1
200 21438 http://www.site.com.br "Mozilla/4.0 (compatible; MSIE 8.0; Windows
NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729;
.NET CLR 3.0.30729; Media Center PC 6.0; MAGW; AskTbATU2/5.13.1.18107; .NET4.0C)"
```

Esse registro informa que no dia 01 de dezembro de 2012 às 00h00min, do horário local do Brasil, o cliente com o endereço IP 187.115.4.4 requisitou ao servidor o endereço /busca/TV. O servidor foi informado que o cliente suporta HTTP/1.1. O servidor respondeu essa requisição com um código de sucesso (indicado pelo código 200) e transferiu 21438 bytes de conteúdo para o cliente. O endereço onde o cliente encontrava-se no momento da requisição era http://www.site.com.br e o navegador utilizado pelo cliente era o Internet Explorer 8.0 (MSIE 8.0) no SO Windows 7 (versão 6.1).

A Tabela 5.2 resume as estatísticas de acesso retiradas do *site* de *e-commerce*. No total, mais de 31 milhões de acessos foram recebidos pelo *site* durante o período de análise e mais de 631 GB de dados foram enviados para os clientes. O *site* recebeu em média 242 requisições por minuto e enviou para os clientes em média 4,99 MB de dados por minuto.

TAB. 5.2: Resumo dos registros coletados.

Duração	12 Dez, 2012 - 28 Fev, 2013
Total de requisições	31.355.585
Média de requisições/minuto	242
Total de bytes transferidos (GB)	631,43
Média de bytes transferidos/minuto (MB)	4,99

5.2.2 CARGA DE TRABALHO WEB

Essa seção apresenta os resultados da caracterização da carga de trabalho Web do *site* de *e-commerce* hospedado na nuvem da Amazon.

A primeira análise efetuada nesse trabalho representa a versão do protocolo HTTP (*HyperText Transfer Protocol*) suportada pelos clientes nas requisições. A Tabela 5.3

apresenta os resultados da análise. Como esperado, HTTP/1.1 é utilizado pela maioria dos clientes, sendo responsável por 99,98% das requisições. Entretanto, é visto que ainda existem clientes que utilizam o HTTP/1.0, apesar de representar uma pequena quantidade de acessos, com apenas 0.2%. Isso indica que o suporte a versão HTTP/1.0 está ficando escasso e a utilização da versão HTTP/1.1 é predominante na Web. Foram encontradas cinco requisições nos registros de acessos que não possuíam uma versão HTTP válida. Esses erros foram ignorados por não terem efeitos significantes nos resultados.

TAB. 5.3: Composição das versões HTTP suportadas pelo cliente.

Versão HTTP	**% de requisições**	**% de Dados Transferidos**
1.0	0,02	0,05
1.1	99,98	99,95
Total	**100,00**	**100,00**

A próxima análise realizada foram os métodos incluídos em cada requisição. A Tabela 5.4 mostra a distribuição das requisições organizadas pelos métodos HTTP. De todas as requisições, 94,41% são do tipo GET, indicando que o endereço da URL é simples de ser recuperado (FIELDING, 1999). Nessa categoria estão incluídos os "GETs condicionais" (requisições que incluem o cabeçalho HTTP *If-Modified-Since*) e os "GETs parciais" (requisições que incluem o cabeçalho HTTP *Range*) (FIELDING, 1999). Infelizmente não há informações nos registros de acesso que determinam a quantidade exata de cada um dos tipos de GET. Os outros dois métodos mais comuns encontrados são POST e HEAD. Requisições do tipo HEAD são feitas quando o conteúdo do arquivo não é necessário e apenas o cabeçalho HTTP do mesmo é transferido. Requisições do tipo POST indicam que o cliente enviou informações ao servidor que não estavam contidas na URL.

TAB. 5.4: Distribuição dos métodos HTTP pelo cliente.

Método	**% de requisições**	**% de Dados Transferidos**
GET	94,41	95,18
POST	5,53	4,81
HEAD	0,05	0,000009
Outros	0,01	0.009991
Total	**100,00**	**100,00**

A Tabela 5.5 mostra a composição dos códigos de resposta HTTP enviados pelo servidor. As descrições completas dos códigos de resposta HTTP estão descritas na especi-

ficação do protocolo HTTP/1.1 (FIELDING, 1999). Conforme mostrado, a maioria das requisições são feitas com sucesso (*status* da resposta 200). As transferências que obtiveram sucesso em sua resposta chegam a 78,04%, entretanto, muitos erros ocorreram, principalmente erros do cliente. Dentre todos os códigos de erro, o mais comum foi o 404 (*File Not Found*) provenientes de arquivos não encontrados no servidor.

TAB. 5.5: Resumo dos códigos de resposta enviados pelo servidor.

Código de Resposta	**% de requisições**	**% de Dados Transferidos**
200 (Sucesso)	78,04	87,13
3xx (Redirecionamentos)	4,10	0,12
4xx (Erros do cliente)	17,01	12,72
5xx (Erros do servidor)	0,80	0,01
Outros	0,05	0,02
Total	**100,00**	**100,00**

Outra análise efetuada foi quanto ao cabeçalho HTTP *http_referer* das requisições. Essa informação é importante, pois nos mostra a relação do endereço de origem do cliente no momento da requisição. A Tabela 5.6 mostra que 66,93% das requisições foram feitas do próprio *site*, enquanto 26,58% das requisições não tiveram o cabeçalho HTTP *http_referer* preenchidos.

TAB. 5.6: Resumo dos cabeçalhos HTTP *http_referer* enviados pelo cliente.

Endereço de Origem	**% de requisições**	**% de Dados Transferidos**
Próprio *site*	66,93	64,84
Vazio	26,58	26,96
Google	5,52	7,42
Bing	0,16	0,20
Yahoo	0,14	0,15
Facebook	0,02	0,02
Outros	0,65	0,41
Total	**100,00**	**100,00**

A última análise feita examina os clientes únicos que acessaram o *site* de *e-commerce* durante o período analisado. Determinar um valor exato de clientes é praticamente impossível, dada a presença de *proxies* e estações de trabalho compartilhadas. A utilização de IPs dinâmicos também aumenta o número de IPs únicos que acessaram o *site* pois um cliente pode ter acessado o *site* utilizando vários IPs de saída. Mesmo com essas restrições, calculamos que os registros de acesso ao servidor contém 1.395.001 endereços de IP

únicos, ou seja, a média de requisições feitas por cada cliente foi de aproximadamente 22 requisições por cliente.

5.2.3 ANÁLISE DE UTILIZAÇÃO

A Figura 5.2 mostra o volume de tráfego diário tratado pelo *site* de *e-commerce* durante o período de monitoramento.

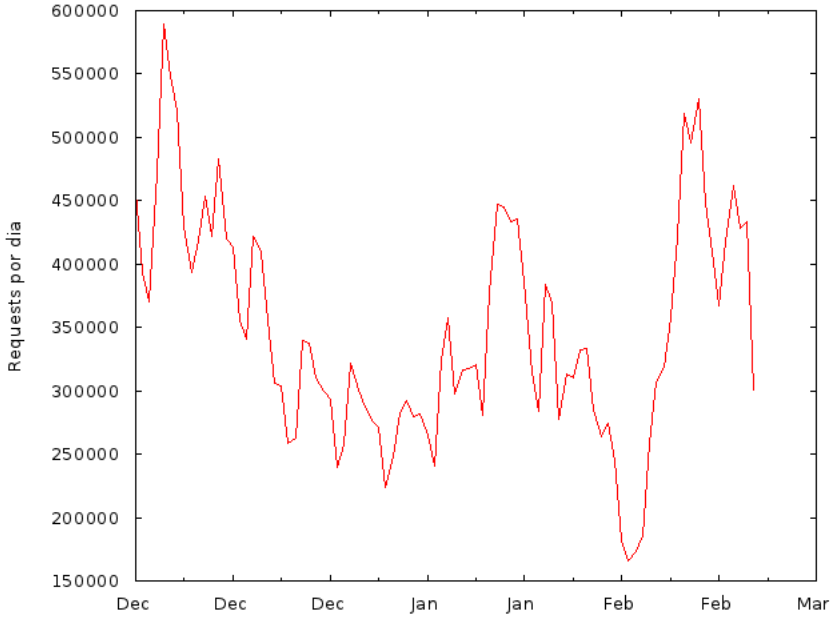

FIG. 5.2: Volume de tráfego diário do *site* de *e-commerce*.

Durante a primeira quinzena do mês de dezembro, o volume de tráfego no *site* é grande devido às vendas para o Natal. Do final de dezembro até a metade de janeiro, o nível de acessos é reduzido, pois esse período marca as programações de final e início de ano que, geralmente, são menos atrativas para o comércio eletrônico. Após a primeira quinzena de janeiro, começando no dia 15, o volume de tráfego do *site* cresce repentinamente. Isto marca o início de liquidações e o *site* se torna muito popular e permanece popular por um período de tempo, até o final do mês. Na primeira quinzena do mês de fevereiro, o volume de tráfego é bastante leve pois coincide com o carnaval, no dia 12, embora o final do mês mostre o início de grandes quantidades de acesso em antecipação as vendas da páscoa. No dia 14 de fevereiro, o *site* volta a passar das 300 mil requisições por dia e permanece com grandes quantidades de acesso até o final do mês. No dia 20 de fevereiro o *site* recebe mais de 530 mil pedidos, embora o dia mais movimentado para o *site* no

período analisado, foi 5 de dezembro, quando mais de 589 mil pedidos foram manipulados pelo *site* de comércio eletrônico.

Para compreender melhor o comportamento dos acessos ao *site*, analisamos o tráfego em maiores detalhes. A Figura 5.3 mostra o volume de tráfego por hora do *site* de *e-commerce*.

A Figura 5.3 consiste em treze gráficos, um para cada semana durante o período analisado. A curva em vermelho de cada gráfico representa o volume de requisições por hora (eixo y) pelo tempo de cada requisição (eixo x). A escala de ambos os eixos são constantes entre todos os gráficos para facilitar a comparação do volume de tráfego ao passar do tempo e para cada dia da semana. As datas comemorativas do Brasil também estão listadas no gráfico para acompanharmos o volume de tráfego antes, durante e após essas datas.

A Figura 5.3 também revela que existem muitas variáveis que afetam o volume de tráfego por hora do *site* de *e-commerce*. Por exemplo, o volume de tráfego aumenta consideravelmente após às 12h00min. Isso acontece pois a maioria das pessoas possuem esse horário livre para o almoço e podem utilizá-lo para navegar na Internet e, inclusive, efetuar compras *online*. Esses acessos representam um fenômeno de *Flash Crowd* em pequena escala. Podemos visualizar *Flash Crowds* em maior escala nos dias 7 de dezembro e 15 de janeiro, sendo eles os picos de acesso, com mais de 40 mil requisições por hora.

Uma outra observação interessante na Figura 5.3 é que o volume de tráfego do *site* de *e-commerce* durante o final de semana é, em geral, um pouco menor do que durante os dias da semana. Uma razão para isso é nos finais de semana as pessoas preferem ir às lojas físicas ao invés de comprarem pela Internet. Quando as pessoas não podem ir às lojas, como em dias de trabalho ou estudo, elas vão à Web procurar e comprar produtos. Podemos perceber também que nas segundas-feiras há um aumento significativo na quantidade de acessos e, historicamente, na quantidade de vendas do *site*.

5.3 METODOLOGIA DOS EXPERIMENTOS

Para a realização dos testes em um ambiente real, o sistema precisou ser parcialmente implementado com os componentes mínimos de funcionamento. Em um primeiro momento, houve a necessidade de se armazenar algumas informações fundamentais para a geração dos resultados preliminares. É importante ressaltar que a arquitetura proposta pode funcionar em tempo real, porém nesta fase inicial, apenas as funcionalidades básicas

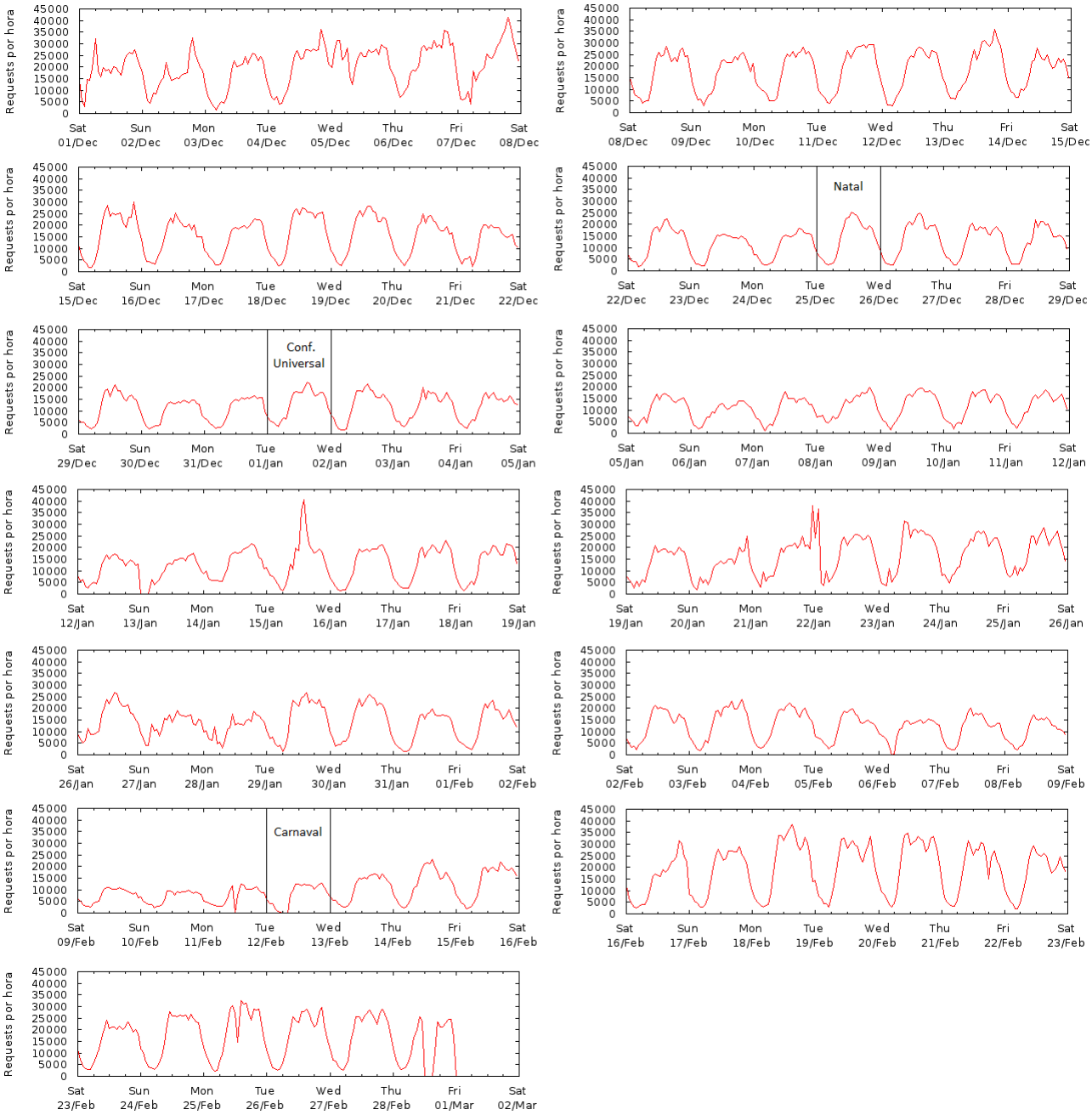

FIG. 5.3: Volume de tráfego por hora do *site* de *e-commerce*.

do sistema foram implementadas e os resultados foram gerados posteriormente através de *scripts* que processam as informações armazenadas pelo sistema. A Figura 5.4 ilustra os componentes do sistema que foram implementados para a obtenção dos resultados preliminares.

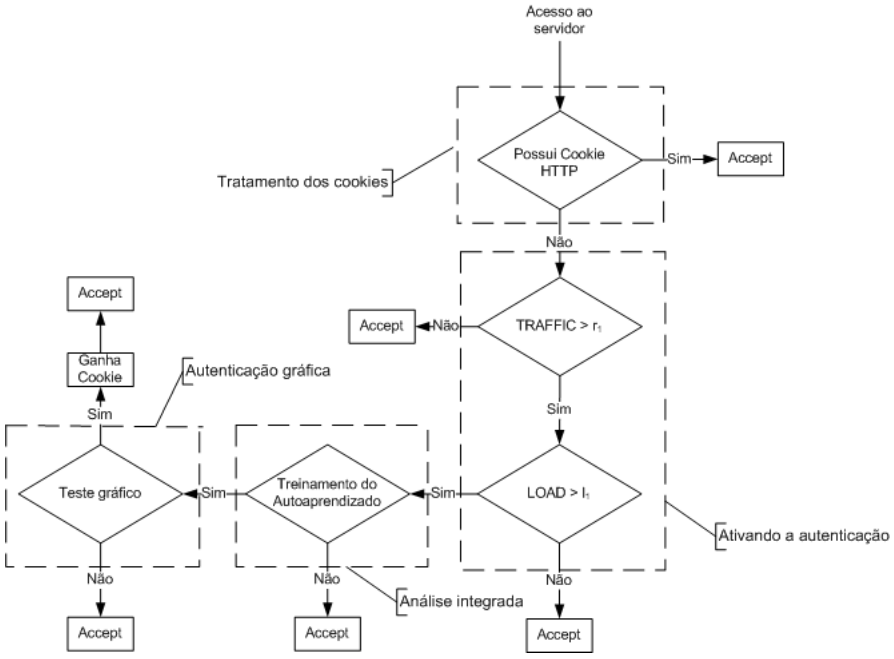

FIG. 5.4: Fluxo básico do sistema para os experimentos.

Após o acesso ao servidor na primeira fase, é feito o tratamento dos *cookies*. Basicamente é verificado se o cliente possui o *cookie* HTTP que o libera dos testes. Caso ele possua, seu acesso é permitido diretamente e a autenticação gráfica não será solicitada. A segunda fase é responsável pela ativação da autenticação. Nessa fase, é verificado se a quantidade de acessos ao servidor passou de um limite aceitável e se a carga de trabalho do servidor é alta. Caso essas condições sejam verdadeiras, o servidor entra na terceira fase, a análise integrada. Nessa fase, o sistema verifica se o autoaprendizado está em modo treinamento ou utilização. No modo utilização o sistema permite que o acesso seja feito sem qualquer interferência. No modo treinamento o sistema passa para a quarta e última fase, chamada autenticação gráfica. Nessa fase, o sistema solicita ao usuário que ele responda um teste gráfico, conhecido como CAPTCHA. Independente da resposta do cliente, mesmo se for nula, o acesso é liberado e o cliente visualiza a página desejada. Caso o cliente responda corretamente à autenticação gráfica, o sistema atribui um *cookie* ao usuário e o mesmo não será mais solicitado à autenticação.

Como visto anteriormente, independente de qualquer situação, o sistema implemen-

tado para os experimentos não bloqueia acesso algum. Ele foi utilizado apenas para armazenar as respostas dos usuários, que servirão de base para a geração dos resultados preliminares. Além disso, o sistema também armazena o nível de relevância da URL acessada através do *PageRank* do Google e o nível de modificação do *site* através do cabeçalho HTTP *Last-Modified*. A Tabela 5.7 mostra a estrutura das informações que foram armazenadas durante a fase de experimentação do sistema.

TAB. 5.7: Estrutura das informações armazenadas no experimento.

IP do cliente	**Data/hora**	**URL**	**PageRank**	**Modificação**	**Resposta**
200.222.88.90	15/Jan/2013:14:00:33	/Eletrodomesticos/1062/	Muito baixa	Médio	Certa
187.108.134.210	15/Jan/2013:14:00:36	/	Média	Médio	Errada
201.29.109.54	15/Jan/2013:14:00:36	/busca/rede+de+ping+pong	Muito baixa	Muito antigo	Certa

Após o armazenamento das informações, foi efetuada a manipulação dos dados através de *scripts* automatizados criados com o objetivo de simular o autoaprendizado do sistema nos períodos de pico de acessos e alta carga de trabalho do servidor Web. Para isto, foi utilizado um algoritmo que efetua os mesmos passos para os diversos períodos de testes obtidos no experimento. O fluxo desse algoritmo pode ser observado na Figura 5.5.

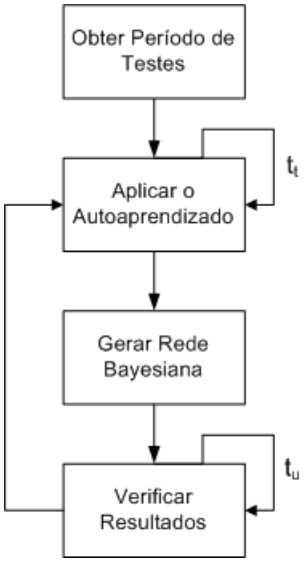

FIG. 5.5: Fluxo do algoritmo para geração dos resultados.

Primeiramente, definiu-se as métricas de pontuação para o autoaprendizado. A cada quatro acertos na resposta do teste gráfico, foi aplicada uma recompensa de 0,1 pontos percentuais a favor da ocorrência de um fenômeno de *Flash Crowd* para o tipo de *site* acessado. Já a regra de punição do autoaprendizado foi configurada para que, a cada dois

erros na resposta do teste gráfico, fosse aplicada um punição de 0,1 pontos percentuais a favor da ocorrência de um ataque DDoS Web para aquele acesso. Esses valores foram arbitrados para a obtenção dos resultados e isso permite que ajustes possam ser feitos com a experiência e características de uso de cada *site*.

Após a configuração do autoaprendizado, foram obtidos os períodos de testes nos quais o servidor passou por picos de acessos com uma alta carga de trabalho. O tamanho desses períodos é variável, podendo possuir diferentes tempos uns dos outros. Após isto, o *script* automatizado faz uma análise dos registros e aplica o autoaprendizado seguindo as regras propostas anteriormente. O tempo de treinamento do autoaprendizado foi definido em $t_t = 10$ minutos e o tempo de utilização do sistema em $t_u = 10$ minutos. Dessa forma, assim que o autoaprendizado é ativado, a fase de treinamento é iniciada com uma duração de 10 minutos. Após isto o sistema passa a utilizar as probabilidades aprendidas durante os próximos 10 minutos, voltando para o treinamento ao final. A cada fase de treinamento, o *script* gera uma nova rede bayesiana com as novas probabilidades para posteriormente, na fase "Verificar Resultados", analisar se os registros da fase de utilização foram classificados como *Flash Crowds* ou ataques DDoS.

Na última fase, o *script* utiliza as novas tabelas de probabilidade para definir o tipo de anomalia encontrada. De acordo com a classificação da anomalia, o *script* marca o acesso como bloqueado ou liberado. Durante os períodos de testes, foram utilizados alguns robôs previamente configurados para navegar no *site* com determinados IPs conhecidos. Esses robôs simulam um acesso malicioso, sendo usados para a análise da eficiência da técnica proposta. O ideal é que a maior parte dos acessos oriundos desses IPs seja classificado como ataques DDoS Web e que a maior parte dos acessos legítimos seja classificado como *Flash Crowds*. Para os experimentos, foi assumido que existiam apenas estes robôs acessando o *site* junto aos usuários legítimos, não existindo outras ações maliciosas no período de avaliação.

5.4 RESULTADOS

Essa seção tem por objetivo demonstrar e analisar os resultados gerados a partir da simulação do sistema em um ambiente real. As experimentações foram realizadas durante um período de 20 dias, de 13 de janeiro até 01 de fevereiro de 2013, e os dados coletados foram analisados e serão comentados a seguir.

Para facilitar a demonstração dos resultados, será utilizada uma matriz de confusão

(MAROM, 2010) para a visualização dos resultados. Uma matriz de confusão é simplesmente uma matriz quadrada que indica as classificações corretas e incorretas de uma determinada classe. O nome deriva da facilidade de se visualizar as confusões que o sistema faz entre duas classes, ou seja, quando o mesmo classifica uma classe equivocadamente como a outra. A classe que está sendo analisada aparece nas linhas da matriz e as classificações encontradas aparecem nas colunas. A diagonal da matriz corresponde às classificações corretas.

A seguir são demonstrados os resultados de quatro experimentos realizados em diferentes períodos de utilização do *site*. Para cada experimento, exceto o primeiro, as probabilidades associadas a ocorrência das anomalias passaram por um ou mais períodos de treinamento do autoaprendizado, nos quais tiveram suas probabilidades adaptadas de acordo com as características dos acessos realizados. Vale ressaltar que o *site* em questão não possui páginas com um *PageRank* do Google com um valor acima de quatro e, por isto, as probabilidades vinculadas aos níveis de relevância muito alta e alta não foram alteradas e não serão demonstradas.

5.4.1 EXPERIMENTO 01

Para exemplificar a utilização da matriz de confusão nos resultados, foi efetuada uma simulação com 10 minutos de duração e um total de 8342 requisições. Para esse experimento, foram utilizadas as probabilidades a priori demonstradas na metodologia deste trabalho, na Tabela 3.1. Esse experimento foi efetuado sem utilizar a autenticação gráfica, se baseando apenas nas probabilidades a priori. A matriz de confusão a seguir representa os resultados do experimento.

	$DDoSWeb$	*Flash Crowd*
$DDoSWeb$	32%	68%
Flash Crowd	23,5%	76,5%

A diagonal da matriz corresponde aos acertos na detecção das anomalias. Para os acessos maliciosos, percebe-se que apenas 32% dos mesmos foram classificados como ataques DDoS Web. Já para os acessos legítimos, 76,5% foram classificados como *Flash Crowds* e tiveram seus acessos ao *site* liberados. As outras informações da matriz demonstram os erros do sistema, que classificou 68% dos acessos maliciosos como *Flash Crowds* e 23,5% dos acessos legítimos como ataques DDoS Web. Podemos observar que as probabilidades

a priori não são suficientes para diferenciar com precisão as anomalias e que sem o autoaprendizado a eficiência da técnica proposta não é satisfatória. O objetivo dos próximos experimentos é utilizar um tempo de treinamento para que as anomalias sejam diferenciadas com mais precisão, fazendo com que a diagonal da matriz de confusão se aproxime ao máximo de 100%, que seria o resultado ótimo na detecção das anomalias.

5.4.2 EXPERIMENTO 02

No segundo experimento, foi utilizado um período de pico de acessos com duração de 15 minutos. Esse período foi escolhido inicialmente por ser um período menor, no qual foram simulados 10 minutos de autoaprendizado das probabilidades e 5 minutos de utilização do sistema. Ao todo, foram 7380 requisições para o treinamento do autoaprendizado e 3350 acessos no período de utilização do sistema. A Tabela 5.8 ilustra as probabilidades adaptadas após o primeiro e único período de treinamento.

TAB. 5.8: Probabilidades após a fase de treinamento do experimento 02.

Modificação do *Site*	**Relevância do *Site***	***Flash Crowd***	**DDoS**
Muito recente	Média	87,9%	12,1%
Muito recente	Baixa	67,3%	32,7%
Muito recente	Muito baixa	74,1%	25,9%
Recente	Média	71,3%	28,7%
Recente	Baixa	71,1%	28,9%
Recente	Muito baixa	53,9%	46,1%
Médio	Média	43,6%	56,4%
Médio	Baixa	45%	55%
Médio	Muito baixa	50,5%	49,5%
Antigo	Média	37,7%	62,3%
Antigo	Baixa	33,4%	66,6%
Antigo	Muito baixa	17,8%	82,2%
Muito antigo	Média	20%	80%
Muito antigo	Baixa	22,3%	77,7%
Muito antigo	Muito baixa	5,2%	94,8%

Após o treinamento das probabilidades, os acessos posteriores ao autoaprendizado foram classificados de acordo com a anomalia encontrada. A matriz de confusão a seguir foi gerada a partir dessa análise.

	DDoSWeb	*Flash Crowd*
DDoSWeb	37,8%	62,2%
Flash Crowd	25,3%	74,7%

Pode-se perceber nesse experimento que, mesmo com 10 minutos de autoaprendizado, o sistema só foi capaz de bloquear 37,8% dos acessos maliciosos, efetuados através de robôs. Além disso, o sistema bloqueou 25,3% dos acessos ditos legítimos, demonstrando que, nesse período de acessos, o autoaprendizado não foi eficiente para otimizar a detecção das anomalias. Apesar dos resultados terem sido melhores do que os resultados do experimento anterior, o curto tempo de treinamento não foi capaz de adaptar com eficiência as probabilidades de ocorrência das anomalias, gerando muitos falso-negativos para os ataques DDoS Web.

5.4.3 EXPERIMENTO 03

O terceiro experimento foi efetuado num período maior do que o segundo, com 36 minutos de duração. Nesse período, o autoaprendizado foi aplicado em duas etapas, nos tempos de 0 a 10 minutos e de 20 a 30 minutos, totalizando 20 minutos de treinamento das probabilidades com 14105 solicitações de autenticação gráfica. Por sua vez, a utilização do sistema deu-se nos tempos de 10 a 20 minutos e de 30 a 36 minutos, dando um total de 16 minutos de utilização do sistema e 10421 requisições. Somando-se os acessos no período de treinamento e utilização do sistema, durante todo o experimento, o site recebeu 24526 requisições. As probabilidades ajustadas após os dois períodos de autoaprendizado serão demonstradas na Tabela 5.9.

Nesse experimento, as probabilidades foram ajustadas e três classificações foram alteradas. Para as páginas com modificação muito recente ou recente, e baixa relevância, os acessos foram classificados como ataques DDoS Web. Já para as páginas com modificação antigo e relevância baixa, os acessos foram classificados como *Flash Crowds*. A matriz de confusão a seguir ilustra os resultados obtidos nesse experimento após as fases de utilização do sistema.

	DDoSWeb	*Flash Crowd*
DDoSWeb	83,9%	16,1%
Flash Crowd	22,5%	77,5%

É possível observar que com um tempo maior de autoaprendizado, os resultados con-

TAB. 5.9: Probabilidades após as fases de treinamento do experimento 03.

Modificação do *Site*	**Relevância do *Site***	***Flash Crowd***	**DDoS**
Muito recente	Média	95,1%	4,9%
Muito recente	Baixa	73,2%	26,8%
Muito recente	Muito baixa	49,4%	50,6%
Recente	Média	98,3%	1,7%
Recente	Baixa	76,4%	23,6%
Recente	Muito baixa	49,2%	50,8%
Médio	Média	61,6%	38,4%
Médio	Baixa	45%	55%
Médio	Muito baixa	47,5%	52,5%
Antigo	Média	33,8%	66,2%
Antigo	Baixa	55,9%	44,1%
Antigo	Muito baixa	20%	80%
Muito antigo	Média	20%	80%
Muito antigo	Baixa	22%	78%
Muito antigo	Muito baixa	11,2%	88,8%

vergem positivamente, visto que quanto maior os valores da diagonal da matriz, melhor foi a detecção das anomalias. Percebe-se que nesse período de simulação, os acessos maliciosos foram classificados em sua maioria como ataques DDoS Web, totalizando 83.9% dos acessos com essa classificação. É interessante observar que a detecção de *Flash Crowds* foi maior do que a do experimento anterior, demonstrando que o sistema reagiu bem a um período maior de treinamento. Um problema deste experimento é que, mesmo com um maior acerto na classificação dos acessos maliciosos, ainda sim houve 22,5% de acessos legítimos classificados como ataques DDoS. Esse número ainda é alto tendo em vista que aproximadamente 1 em cada 5 acessos legítimos seriam bloqueados.

5.4.4 EXPERIMENTO 04

Neste último experimento, foi utilizado um tempo de duração maior do que nos experimentos anteriores, com 51 minutos de duração. Isso fez com que o sistema tivesse 30 minutos de autoaprendizado das probabilidades e 21 minutos de utilização. Durante o treinamento do autoaprendizado, o site recebeu 20967 acessos. Já na utilização do sistema, foram contabilizadas 13873 requisições ao site. A Tabela 5.10 ilustra as probabilidades adaptadas após as fases de treinamento do sistema.

Nesse caso, houve o ajuste de três classificações das anomalias: para páginas recentes

TAB. 5.10: Probabilidades após as fases de treinamento do experimento 04.

Modificação do *Site*	**Relevância do *Site***	***Flash Crowd***	**DDoS**
Muito recente	Média	89,3%	10,7%
Muito recente	Baixa	77,5%	22,5%
Muito recente	Muito baixa	53,2%	46,8%
Recente	Média	96,1%	3,9%
Recente	Baixa	69,8%	30,2%
Recente	Muito baixa	43,9%	56,1%
Médio	Média	73,7%	26,3%
Médio	Baixa	45%	55%
Médio	Muito baixa	43,4%	56,6%
Antigo	Média	45,8%	54,2%
Antigo	Baixa	57,6%	42,4%
Antigo	Muito baixa	32,8%	67,2%
Muito antigo	Média	20%	80%
Muito antigo	Baixa	51,3%	48,7%
Muito antigo	Muito baixa	15,5%	84,5%

e com a relevância muito baixa a anomalia foi classificada como um ataque DDoS Web. Já para páginas com a relevância baixa e níveis de modificação antigo e muito antigo, os acessos foram classificados como um fenômeno de *Flash Crowd*. Para ilustrar os resultados alcançados com este experimento, será utilizada a matriz de confusão a seguir.

	$DDoSWeb$	*Flash Crowd*
$DDoSWeb$	$81,7\%$	$18,3\%$
Flash Crowd	$14,8\%$	$85,2\%$

Esse experimento é válido para observar que, com um tempo maior, os acertos foram maiores que 80% para a detecção de ambas as anomalias. Apesar de detectar menos acessos maliciosos como ataques de DDoS Web do que o experimento anterior, nesse caso pode-se reparar que os acertos na detecção de *Flash Crowds* foram superiores, chegando a 85,2%. É importante considerar que, mesmo com uma menor classificação de ataques DDoS (81,7%), o sistema indica uma quantidade menor de falso-positivos para os *Flash Crowds* (14,8%). Em outras palavras, uma quantidade pequena de acessos legítimos é classificada como ataques DDoS, não chegando a 15% dos acessos.

Podemos concluir com os experimentos realizados que, sem a fase de treinamento, o sistema não é eficiente para a diferenciação das anomalias, produzindo poucos acertos e

muitos erros na classificação das mesmas. Também podemos afirmar que tempos muito curtos de autoaprendizado podem melhorar a diferenciação das anomalias, porém ainda assim classificam muitos acessos legítimos como ataques DDoS Web. Por último, podemos destacar que quanto maior é o tempo de autoaprendizado das probabilidades, melhores são os resultados obtidos a partir da utilização do sistema.

6 CONSIDERAÇÕES FINAIS

Ambos os problemas causados por ataques DDoS Web e fenômenos de *Flash Crowd* consistem num aumento do número de solicitações a um determinado *site* da Web. No entanto, essas anomalias estão sendo cada vez mais parecidas e os seus efeitos vêm causando prejuízos financeiros significativos a cada ataque. Por isso, é importante que existam maneiras eficazes para se distinguir estas duas anomalias.

Nesse trabalho foi proposta uma nova metodologia para diferenciação de ataques DDoS Web e fenômenos de *Flash Crowd* através de uma análise integrada do tráfego de rede e das características dos *sites* vítimas de tais fenômenos. Para essa análise foram utilizadas duas características dos *sites* Web: data de modificação e relevância. Juntamente com essas informações, foi proposto um modelo probabilístico baseado em redes bayesianas para calcular as probabilidades de acontecimento desses fenômenos. Para esse modelo probabilístico é sugerido um autoaprendizado baseado em uma autenticação gráfica que possibilita ao sistema adquirir a capacidade de reconhecimento das anomalias conforme a sua utilização.

A autenticação gráfica consiste em testes gráficos que detectam acessos maliciosos feitos por robôs. Essa autenticação é feita por meio de um CAPTCHA que solicita ao usuário a resolução de um teste gráfico. Clientes legítimos podem facilmente resolver esses testes, enquanto robôs não. A cada erro ou acerto o sistema bloqueia ou libera os acessos e reforça a classificação das anomalias através de recompensas e punições efetuadas no modelo bayesiano. Esse modelo produzirá uma nova rede bayesiana que será utilizada para diferenciar as anomalias, mantendo o sistema em ciclos de utilização.

Para a implementação da metodologia proposta, foi criada uma arquitetura conceitual baseada em camadas. Essa arquitetura foi dividida em camada de sensoriamento, camada de processamento e camada de serviço, sendo cada camada responsável por uma função específica, fornecendo informações umas às outras. Também foram demonstradas as ferramentas necessárias para a implementação do sistema, bem como o motivo de suas adoções. O desenvolvimento do sistema foi feito à partir de um *proxy* reverso com a implementação de uma aplicação denominada StopBots. O StopBots é o centro da arquitetura e possui toda a inteligência necessária ao funcionamento do sistema. Em um primeiro momento, o

sistema foi aplicado a um *site* de comércio eletrônico na Web. Para isto, foi realizada uma análise detalhada da usabilidade do *site*, bem como dos perfis dos usuários que o acessavam e do impacto que a implementação do sistema teria sobre as vendas do *site*. Após isto, os parâmetros necessários ao sistema foram adequados de acordo com as características do *site* e da infraestrutura responsável pelo aprovisionamento do mesmo.

Por fim, foi desenvolvida uma metodologia para validar a técnica proposta através de alguns experimentos em um ambiente real. Essa escolha se deu em função da dificuldade de simulação de fenômenos de *Flash Crowd* em ambientes de laboratório. Para a realização dos testes, o sistema precisou ser parcialmente implementado com os mínimos componentes necessários para a coleta das informações. Após isto, os dados coletados foram analisados para a geração de resultados parciais e matrizes de confusão foram utilizadas para facilitar a visualização das informações. No melhor caso, o sistema foi capaz de classificar 81,7% dos acessos maliciosos como ataques DDoS Web e 85,2% de acessos legítimos como fenômenos de *Flash Crowd*.

6.1 TRABALHOS FUTUROS

O estudo apresentado neste trabalho deixa alguns desafios futuros. A análise das características dos *sites*-alvo é feita baseada em duas informações: relevância e data de modificação da página solicitada. No entanto, é importante que outras características possam ser utilizadas para servir como base para o modelo bayesiano proposto e melhorar a precisão na diferenciação das anomalias. Podemos utilizar, por exemplo, o endereço do *site* de origem da requisição como uma característica da página. Essa informação será útil para analisar de onde os acessos são provenientes. Quando um *site* sofre uma grande quantidade de acessos, estes podem ocorrer por uma publicidade em um *site* popular e, quando os acessos forem provenientes de um mesmo *site*, pode-se concluir que um fenômeno de *Flash Crowd* tem maiores chances de estar ocorrendo.

Outro trabalho futuro será encontrar um método para otimizar os valores das métricas de pontuação do autoaprendizado. Dessa maneira, teremos uma melhor adaptação às mudanças da rede bayesiana de acordo com os erros e acertos descobertos na detecção das anomalias. Esses valores podem, inclusive, ser dinâmicos e se adaptarem ao sistema em tempo real. Outro fator importante para o autoaprendizado será determinar o melhor tempo de treinamento e utilização do sistema para cada caso específico, visto que esses tempos foram fundamentais para a melhoria dos resultados apresentados.

Para viabilizar a solução proposta neste trabalho, um próximo passo será a implementação completa do sistema. Nos experimentos, a técnica foi parcialmente implementada com os componentes mínimos para demonstrar seu funcionamento e oferecer uma prova de conceito. Um possível trabalho futuro inclui a execução em tempo real da proposta. Além disto, será necessário estudar detalhadamente cada parâmetro utilizado no fluxo de funcionamento do sistema, buscando os melhores valores de tempos e limites para a solicitação da autenticação gráfica. Uma alternativa seria ter esses parâmetros adaptados continuamente às características de usabilidade do *site* e desempenho do servidor Web, responsável pela hospedagem da aplicação Web.

Por fim, outro desafio futuro será a aplicação da técnica proposta em outros tipos de *sites*, com características diferentes. Dessa maneira, será possível observar se os resultados demonstrados neste trabalho serão semelhantes aos resultados obtidos de serviços Web que possuem outras características de uso, como *sites* de notícias, blogs e redes sociais.

7 REFERÊNCIAS BIBLIOGRÁFICAS

e-bit: Informações de Comércio Eletrônico. URL `http://www.ebitempresa.com.br/`.

Logging in W3C httpd. URL `http://www.w3.org/Daemon/User/Config/Logging.html`.

BEALE, J. e CASWELL. ***Snort 2.1 Intrusion Detection, Second Edition***. Syngress, 2nd edition, maio 2004. ISBN 1931836043.

BLOOM, B. H. **Space/time trade-offs in hash coding with allowable errors**. *Commun. ACM*, 13(7):422–426, julho 1970. ISSN 0001-0782. URL `http://doi.acm.org/10.1145/362686.362692`.

BUSSCHERS, H. **Effectiveness Of Defense Methods Against DDoS Attacks By Anonymous**. volume 16. University of Twente, 2012.

DARWICHE, A. **Bayesian networks**. *Communications of the ACM*, 53:80–90, dezembro 2010. ISSN 0001-0782. ACM ID: 1859227.

FENG, X. X., PENG, Y. e ZHAO, Y. L. **The Analysis of a Botnet Based on HTTP Protocol**. *Advanced Materials Research*, 179-180:575–579, janeiro 2011. ISSN 1662-8985. URL `http://www.scientific.net/AMR.179-180.575`.

FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P. e BERNERS-LEE, T. **RFC 2616, Hypertext Transfer Protocol – HTTP/1.1**, 1999. URL `http://www.rfc.net/rfc2616.html`.

FITZPATRICK, B. **Distributed caching with memcached**. *Linux J.*, 2004(124):5–, agosto 2004. ISSN 1075-3583. URL `http://dl.acm.org/citation.cfm?id=1012889.1012894`.

FUNG, R. e FAVERO, B. D. **Applying Bayesian networks to information retrieval**. *Communications of the ACM*, 38:42–ff., março 1995. ISSN 0001-0782. ACM ID: 203340.

GUNTHER, N. J. **Linux Load Average Revealed**. Em *30th International Computer Measurement Group Conference, December 5-10, 2004, Las Vegas, Nevada, USA, Proceedings*, págs. 149–160. Computer Measurement Group, 2004.

HAVELIWALA, T. **Efficient Computation of PageRank**. http://ilpubs.stanford.edu:8090/386/, 1999. URL `http://ilpubs.stanford.edu:8090/386/`. Efficient Computation of PageRank Taher H. Haveliwala (taherh@db.stanford.edu).

HECKERMAN, D., MAMDANI, A. e WELLMAN, M. P. **Real-world applications of Bayesian networks**. *Communications of the ACM*, 38:24–26, março 1995. ISSN 0001-0782. ACM ID: 203334.

HOLOVATY, A. e KAPLAN-MOSS, J. ***The Definitive Guide to Django: Web Development Done Right (Pro)***. Apress, Berkely, CA, USA, 2007. ISBN 1590597257.

HONG, B., MILLER, E. L., BRANDT, S. A., LONG, D. D. E. e ARI, I. **Managing flash crowds on the Internet**. *11th IEEEACM International Symposium on Modeling Analysis and Simulation of Computer Telecommunications Systems 2003 MASCOTS 2003*, (October):246–249, 2003.

JAMJOOM, H. e SHIN, K. G. **Persistent Dropping: An Efficient Control of Traffic Aggregates**. Em *In Proceedings of ACM SIGCOMM 2003*, págs. 287–297, 2003.

JEYANTHI, N. e IYENGAR, N. C. S. N. **Escape-on-Sight: An Efficient and Scalable Mechanism for Escaping DDoS Attacks in Cloud Computing Environment**. *Cybernetics and Information Technologies*, 13(1):46–60, março 2013. ISSN 1314-4081.

JUNG, J., KRISHNAMURTHY, B. e RABINOVICH, M. **Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites**. *Proceedings of the 11th international conference on World Wide Web*, pág. 293–304, 2002. ACM ID: 511485.

KAELBLING, L. P., LITTMAN, M. L. e MOORE, A. W. **Reinforcement learning: a survey**. *Journal of Artificial Intelligence Research*, 4:237–285, maio 1996. ISSN 1076-9757. URL `http://portal.acm.org/citation.cfm?id=1622737.1622748`. ACM ID: 1622748.

KRISTOL, D. M. **HTTP Cookies: Standards, privacy, and politics**. *ACM Trans. Internet Technol.*, 1(2):151–198, novembro 2001. ISSN 1533-5399. URL `http://doi.acm.org/10.1145/502152.502153`.

LANGVILLE, A. N. e MEYER, C. D. ***Google's PageRank and Beyond: The Science of Search Engine Rankings***. Princeton University Press, julho 2006. ISBN 9780691122021.

LAU, F. e RUBIN, S. H. **Distributed Denial of Service Attacks**. Em *In IEEE International Conference on Systems, Man, and Cybernetics*, págs. 2275–2280, 2000.

LE, M. Z. Q. **Methods of Distinguishing Flash Crowds from Spoofed DoS Attacks**. *Next Generation Internet Networks, 3rd EuroNGI Conference on*, págs. 167–173, 2007.

LI, K., ZHOU, W., LI, P., HAI, J. e LIU, J. **Distinguishing DDoS Attacks from Flash Crowds Using Probability Metrics**. Em *Network and System Security, 2009. NSS '09. Third International Conference on*, volume 0, págs. 9–17. IEEE Computer Society, 2009. ISBN 978-0-7695-3838-9.

LUTZ, M. ***Learning Python***. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2 edition, 2003. ISBN 0596002815.

MAROM, N., ROKACH, L. e SHMILOVICI, A. **Using the confusion matrix for improving ensemble classifiers**. Em *Electrical and Electronics Engineers in Israel (IEEEI), 2010 IEEE 26th Convention of*, págs. 000555–000559, 2010.

MATTHEW, N. e STONES, R. ***Beginning Databases with PostgreSQL: From Novice to Professional***. Apress, 2nd edition, abril 2005. ISBN 9781590594780.

MILSTEAD, J. e FELDMAN, S. **Metadata: Cataloging by Any Other Name**. *Online*, 23(1):24–26,28–31, 1999. ISSN ISSN-0146-5422. URL `http://www.eric.ed.gov/ERICWebPortal/detail?accno=EJ599607`.

MIRKOVIC, J. e REIHER, P. **A taxonomy of DDoS attack and DDoS defense mechanisms**. *ACM SIGCOMM Computer Communication Review*, 34:39–53, abril 2004. ISSN 0146-4833. ACM ID: 997156.

MURTY, J. ***Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB***. O'Reilly Media, Inc., março 2008. ISBN 0596515812.

NIVEN, L. ***The flight of the horse***. Ballantine Books, 1973. URL `http://books.google.com.br/books?id=R8pKAAAAMAAJ`.

NORSYS. **Netica Bayesian Network Software**, 2011. URL `http://www.norsys.com`.

OIKONOMOU, G. e MIRKOVIC, J. **Modeling human behavior for defense against flash-crowd attacks**. Em *Proceedings of the 2009 IEEE international conference on Communications*, ICC'09, pág. 625?630, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-3434-3. URL `http://dl.acm.org/citation.cfm?id=1817271.1817388`.

PAGE, L., BRIN, S., MOTWANI, R. e WINOGRAD, T. **The PageRank Citation Ranking: Bringing Order to the Web.** Technical Report 1999-66, Stanford InfoLab, November 1999. URL `http://ilpubs.stanford.edu:8090/422/`. Previous number = SIDL-WP-1999-0120.

PENG, T., LECKIE, C. e RAMAMOHANARAO, K. **Protection from Distributed Denial of Service Attack Using History-based IP Filtering**. págs. 482–486, 2003.

POURRET, O., NAÏM, P. e MARCOT, B. ***Bayesian Networks: A Practical Guide to Applications***. Wiley, maio 2008. ISBN 9780470060308.

PUTCHALA, S. e AGARWAL, N. **Machine vision: an aid in reverse Turing test**. *AI Soc.*, 26(1):95–101, fevereiro 2011. ISSN 0951-5666. URL `http://dx.doi.org/10.1007/s00146-009-0231-4`.

REESE, W. **Nginx: the high-performance web server and reverse proxy**. *Linux J.*, 2008(173), setembro 2008. ISSN 1075-3583. URL `http://dl.acm.org/citation.cfm?id=1412202.1412204`.

RUSSELL, S. J. e NORVIG, P. ***Artificial Intelligence: A Modern Approach***. Prentice Hall, 1st edition, janeiro 1995. ISBN 0131038052.

SCHUBERT, M., BENNETT, D., GINES, J., HAY, A. e STRAND, J. ***Nagios 3 Enterprise Network Monitoring: Including Plug-Ins and Hardware Devices***. Syngress, 1st edition, junho 2008. ISBN 1597492671.

SUTTON, R. S. e BARTO, A. G. ***Reinforcement Learning: An Introduction***. The MIT Press, março 1998. ISBN 9780262193986.

THAPNGAM, T., YU, S., ZHOU, W. e BELIAKOV, G. **Discriminating DDoS attack traffic from flash crowd through packet arrival patterns**. Em *INFOCOM WKSHPS 2011 : IEEE Conference on Computer Communications Workshops*, págs. 952–957. IEEE, maio 2012. ISBN 9781457702488, 9781457702495. URL `http://dro.deakin.edu.au/view/DU:30042190`.

VON AHN, L., BLUM, M., HOPPER, N. J. e LANGFORD, J. **CAPTCHA: Using Hard AI Problems for Security**. Em *IN PROCEEDINGS OF EUROCRYPT*, págs. 294–311. Springer-Verlag, 2003.

WANG, J., PHAN, R.-W., WHITLEY, J. e PARISH, D. **DDoS attacks traffic and Flash Crowds traffic simulation with a hardware test center platform**. Em *Internet Security (WorldCIS), 2011 World Congress on*, págs. 15–20. IEEE, fevereiro 2011. ISBN 978-1-4244-8879-7.

YU, S., THAPNGAM, T., LIU, J., WEI, S. e ZHOU, W. **Discriminating DDoS Flows from Flash Crowds Using Information Distance**. Em *Network and System Security, International Conference on*, volume 0, págs. 351–356, Los Alamitos, CA, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3838-9.

YU, S., ZHOU, W., JIA, W., GUO, S., XIANG, Y. e TANG, F. **Discriminating DDoS Attacks from Flash Crowds Using Flow Correlation Coefficient**. *IEEE Transactions on Parallel and Distributed Systems*, 23(6):1073–1080, junho 2012. ISSN 1045-9219. URL `http://www.computer.org/csdl/trans/td/2012/06/ttd2012061073-abs.html`.