Uma Arquitetura para Mitigar Ataques DDoS em Serviços Web sob Nuvem

Fernando Gielow, Fernando Bernardelli, Cinara Menegazzo, Nadine Pari, Aldri Santos

[1] Departamento de Informática – Universidade Federal do Paraná
NR2 - Núcleo de Redes Sem Fio e Redes Avançadas – Curitiba – Brasil

{fhgielow,fcbernardelli,cmenegazzo,nelpari,aldri}@inf.ufpr.br

Abstract. *Distributed Denial of Service (DDoS) attacks are often neglected because they cause just a temporary interruption of the system normal functioning. With the advent of paradigms like Cloud Computing, mitigating DDoS attacks giving more resources to the applications has become a feasible alternative, but entails the Economic DDoS problem. This paper presents an architecture to mitigate DDoS attacks against a Cloud hosted application. Such architecture is based on the idea of instantiating a replica of the application - simple operation for a Cloud - and redirecting only authentic queries to the new replica. The proposed architecture does not need to identify the attackers and, even so, it filters only authentic traffic, without extra overhead and potential categorization errors that could arise when trying to identify the clients.*

Resumo. *Ataques de Distributed Denial of Service (DDoS) frequentemente são negligenciados por representarem apenas uma interrupção temporária no funcionamento normal de um sistema. Com o advento de paradigmas como a cloud, a mitigação deste tipo de ameaça com o acréscimo de recursos para as aplicações se torna viável, mas acarreta em um problema denominado economic DDoS. Este artigo apresenta uma proposta de arquitetura para a mitigação de ataques DDoS direcionados a uma aplicação hospedada em uma cloud. Tal arquitetura é baseada na instanciação de uma réplica da aplicação - operação simples em uma cloud - e no redirecionamento apenas de requisições legítimas a esta réplica. A arquitetura proposta não precisa identificar os clientes atacantes e, ainda assim, consegue filtrar apenas o tráfego legítimo sem a carga e possíveis erros decorrentes da necessidade de identificação.*

1. Introdução

Diversas pesquisas têm sido desenvolvidas para tratar questões da Internet atual, que podem se propagar para a Internet do Futuro (IF). Tais problemas podem ser amplamente categorizados nas áreas de mobilidade, qualidade de serviço e segurança, os quais ainda caminham para soluções aceitáveis, agravados pelo surgimento de novas arquiteturas. Hoje tanto os dados quanto as aplicações são oferecidos em localizações físicas distintas e desconhecidas. Outra grande mudança ocorreu na forma de administrar um sistema, que antes era de âmbito mais local, com seus usuários e servidores característicos, e agora esses sistemas são hospedados em ambientes construídos pelo compartilhamento de recursos de diversos sistemas autônomos (AS) e heterogêneos [Pianese et al. 2010].

Apesar de muitos esforços em pesquisas, os ataques de *Denial of Service* (DoS) ainda representam sérias ameaças a muitos servidores na Internet e se configuram como

um dos principais desafios de segurança atualmente propagado para a IF, que interconectará muito mais dispositivos e indivíduos. Um ataque DoS não visa invadir um computador para obter informações confidenciais, nem tão pouco alterar informações armazenadas nele. Seu objetivo é a indisponibilização de um serviço fornecido, utilizando-se do encaminhamento de grandes quantidades de tráfego ao hospedeiro do serviço. Essa questão torna-se ainda mais severa quando diversos geradores de tráfego intensificam o encaminhamento de tráfego de maneira distribuída, caracterizando um ataque de *Distributed Denial of Service* (DDoS) [Sachdeva et al. 2008]. Embora tal carga seja um problema apenas momentâneo, em se tratando de aplicações destinadas ao comércio eletrônico, por exemplo, uma parada do serviço representa grandes perdas financeiras.

Com as novas arquiteturas de rede e de aplicações que configuram a Internet, têm surgido sistemas complexos e robustos como *clouds* (nuvens), onde o desafio de mitigar ataques DoS torna-se ainda mais necessário. Embora a maioria das soluções comumente oferecidas para mitigar DDoS em *cloud* se baseie na maior alocação de recursos [Peng et al. 2007], essas abordagens tornam-se inadequadas pois a premissa da possibilidade de maior alocação de recursos nem sempre é viável por ser custosa demais [Bakshi and Yogesh1 2010], [Liu 2010]. Este comportamento carateriza o *economic DDoS* (eDDoS) [Khor and Nakao 2009].

Este trabalho propõe uma arquitetura reativa e tolerante a falhas para a mitigação de ataques de DDoS executados contra aplicações hospedadas em uma *cloud*. Tal arquitetura é baseada na instanciação de uma réplica da aplicação e no redirecionamento apenas de requisições legítimas a esta réplica. A arquitetura monitora o tráfego de uma aplicação e ao detectar uma possível anomalia, isto é, a ocorrência de um ataque de DDoS, ela estabelece uma nova instância desta aplicação, garantindo que nenhum tráfego malicioso a alcance. As diferenças desta solução para outras propostas são que a aplicação hospedada não precisa prover acurácia na filtragem de tráfego legítimo, o uso dos recursos não é onerado financeiramente, e a intervenção humana é desnecessária. Uma avaliação experimental considerando o tempo de resposta aos clientes mostra a eficácia de uma implementação da arquitetura diante de ataques DDoS a um serviço Web.

O restante do artigo está organizado da seguinte maneira: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 detalha a arquitetura proposta para a mitigação de ataques *DDoS*. A Seção 4 apresenta uma descrição da implementação realizada da arquitetura. A Seção 5 apresenta uma avaliação, juntamente com o cenário e os resultados. Por fim, a conclusão e trabalhos futuros são apresentados na Seção 6.

2. Trabalhos Relacionados

As pesquisas que envolvem propostas de mitigação de DDoS em arquiteturas de *cloud*, ainda são consideradas incipientes e distantes de uma convergência. Dentre as poucas propostas para estes ambientes, destaca-se o *framework* pró-ativo CluB, apresentado em [Hazelhurst 2008], que sugere que sejam selecionados determinados roteadores, dispostos de forma distribuída, para a análise de tráfego e consequente prevenção de atividade maliciosa. Neste *framework*, todo pacote deve ser verificado para entrar, sair ou transitar na arquitetura. Cada roteador alocado realiza a verificação, o que é custoso devido ao *overhead* causado pela autenticação de cada pacote e pela necessidade inviável de alterar o comportamento dos roteadores.

Em [Verkaik et al. 2006], é apresentado um esquema pró-ativo que emprega Comunidades de Interesse (COIs) para capturar dados sobre o comportamento coletivo das entidades remotas, utilizando-os para predizer o comportamento futuro. Tal esquema assume que os clientes que tiveram relações legítimas anteriormente possuem bons indícios e podem ser considerados novamente legítimos. Entretanto, a identificação dos clientes antigos não é tão trivial. Além do *overhead* gerado pela verificação, os endereços IPs são normalmente dinâmicos e a exigência da realização de *login* para a identificação não é possível, dado que o ataque de DDoS pode impossibilitar uma operação de identificação. Em [Bakshi and Yogesh1 2010], os ataques são tratados através da criação de uma nova instância da aplicação. Uma vez que um ataque DDoS é detectado, o mecanismo proposto busca identificar os atacantes através de PINGs: caso um cliente suspeito de ser atacante não responda ao PING, ele é considerado como um atacante, não sendo redirecionado para a nova instância da aplicação. Entretanto, essa solução assume que sempre e apenas clientes genuínos responderão a PINGs, o que as vezes não condiz com a realidade.

A eficácia desses esquemas de mitigação depende diretamente da capacidade de identificação ou filtragem dos clientes legítimos. A solução WebSoS [Stavrou et al. 2005] oferece uma filtragem robusta de tráfego atacante e bloqueio de requisições não aprovadas, formando assim um *overlay* seguro que mitiga DDoS em servidores web. O servidor utiliza mecanismos de autenticação criptográfica e um teste gráfico de Turing [Dietrich et al. 2000] para diferenciar clientes humanos de *scripts* de ataque. Estes procedimentos, segundo os testes dos autores, não sobrecarregam o funcionamento do serviço, porém exigem que os roteadores localizados no perímetro do servidor sejam reconfigurados, o que é inviável para arquiteturas de *cloud*.

3. Arquitetura para Mitigação de DDoS em *Cloud*

Esta Seção descreve uma arquitetura para mitigar ataques de DDoS em *clouds* de forma autônoma e independente. A arquitetura proposta pode ser utilizada por qualquer aplicação web hospedada em uma *cloud* que, ao sofrer indícios de um ataque DDoS, filtra o tráfego legítimo e encaminha apenas este para uma nova instância da mesma aplicação.

Esta arquitetura, ilustrada na Figura 1, é composta por um módulo geral chamado de Gerenciador de Tráfego (GT), que não se comunica diretamente com a aplicação. Esse módulo possui os submódulos INA, GB, AT e RT. Além disso, a instância do banco de dados (BD) é exterior às demais instâncias da *cloud*, visto que o banco de dados também está nas nuvens e pode ser acessado de qualquer outra instância *cloud*.

O submódulo AT observa o comportamento do tráfego de entrada para a aplicação de forma pró-ativa. Ele foca na estimativa da quantidade de tráfego e de processamento no servidor, e realiza medição para detectar a existência de um possível ataque DDoS. Caso um ataque seja detectado, o submódulo INA é ativado. O INA criará uma nova instância da aplicação em outro servidor na *cloud*, consequentemente com um endereço IP diferente. O submódulo RT trata todo o tráfego de entrada, respondendo com um redirecionamento para a nova instância da aplicação, como visto na Figura 2. O RT parte do princípio que os atacantes DDoS não interpretam as respostas obtidas do servidor, pois se eles interpretarem, sua eficiência é reduzida. Desta maneira, apenas os clientes legítimos (Nós *C* na figura) serão, de fato, redirecionados à nova aplicação, enquanto os nós atacantes não passarão pelo RT.

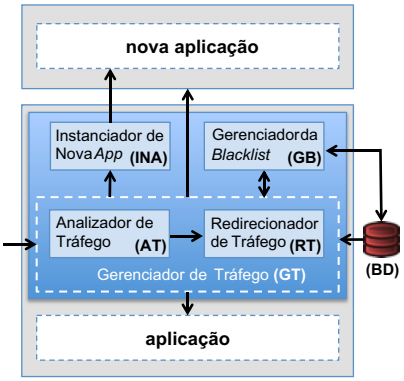

Figura 1. Arquitetura de mitigação de DDoS

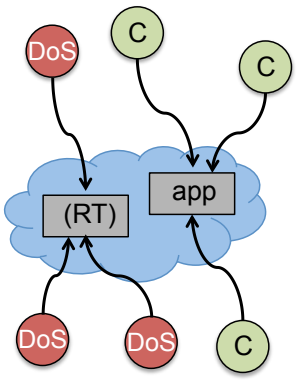

Figura 2. Fluxo de tráfego

Ao tentar redirecionar os clientes para a nova instância, o endereço do cliente, seja ele legítimo ou não, será adicionado em uma *blacklist*. Os clientes presentes nesta lista têm suas requisições descartadas, a fim de reduzir o custo de processamento de respostas no servidor. Entretanto, como o cliente legítimo é informado do redirecionamento antes de seu endereço entrar na *blacklist*, ele terá acesso à esta nova instância replicada e poderá enviar uma nova requisição. Registros com tempo de validade são empregadas nesta *blacklist*, dado que as respostas podem ser perdidas. O tempo de validade na lista aumenta exponencialmente, para diminuir ainda mais a sobrecarga. Cabe ao GB, o papel de adicionar e gerenciar a saída de endereços de clientes à *blacklist*, assim como o tempo de validade da entrada que aumenta exponencialmente.

4. Implementação

Para a implementação da arquitetura, a solução em *cloud* [Heroku 2012] foi utilizada. Ela oferece infra-estrutura como serviço de hospedagem, possibilitando o desenvolvimento em *Ruby on Rails*. A arquitetura do *framework* [RubyOnRails 2012] é completamente baseada no paradigma *Model View Controler* (MVC), facilitando a organização dos módulos de nossa arquitetura. Assim, a estrutura do código escrito em RoR é composta de componentes de Modelo, de Visão e de Controle. Os componentes de **modelo** correspondem aos dados - como eles são armazenados, obtidos, correlacionados. A parte de **visão** corresponde à parte gráfica da aplicação. Finalmente, os **controladores** realizam a manipulação de dados como um todo, e correspondem à parte lógica e funcional do código. Eles funcionam também como uma ponte entre modelo e visão, para que os dados transitem em ambos os sentidos.

Considerando o RoR, o submódulo analizador de tráfego (AT) da arquitetura corresponde a um controlador. Assim, uma requisição à aplicação será interceptada por esse componente de controle, que realizará a medição de estatísticas, e imediatamente acionará o controlador que corresponde ao funcionamento da aplicação em si. Deve-se notar, contudo, que o tempo despendido neste controlador é ínfimo. A Figura 3 apresenta um fluxograma da implementação realizada. Em outras implementações, caso se perceba que o tempo afeta o funcionamento do mecanismo de mitigação, este processamento poderia ainda ser realizado em segundo plano.

Quando o AT detectar a existência de um possível ataque, uma nova instância *cloud* é criada pelo submódulo INA e a aplicação é replicada para esta instância, parali-

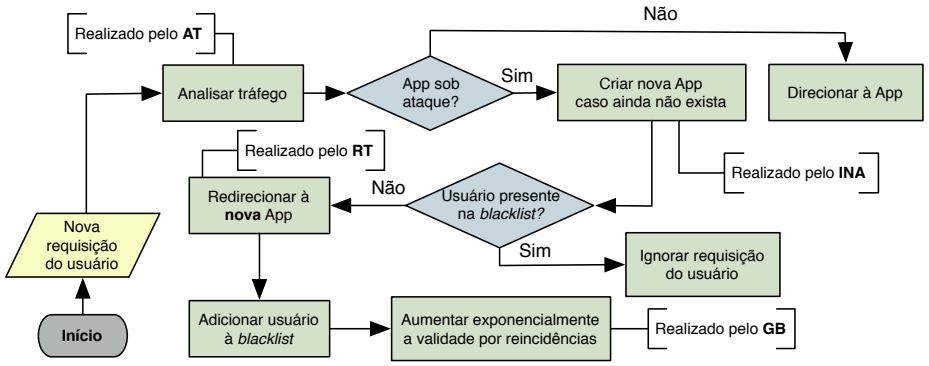

Figura 3. Operações da implementação para a mitigação

sando a aplicação original, que passa a responder apenas como redirecionador. O processo de reinstanciação da aplicação na implementação realizada consiste da existência prévia de uma segunda aplicação, inicialmente sem nenhum recurso alocado.

Uma particularidade interessante do *framework* RoR é a existência de um arquivo de rotas. A implementação do submódulo redirecionador de tráfego (RT) é realizada em cima deste arquivo, chamado *routes.rb*. Para a exibição de qualquer página dinâmica da aplicação, o arquivo de rotas é inevitavelmente chamado. Desta forma, ele é utilizado para a adição de clientes na *blacklist* e respectiva filtragem dos clientes bloqueados pelo gerenciador da *blacklist* (GB). No redirecionamento do tráfego para uma nova instância, uma entrada será adicionada, bloqueando o cliente em questão por determinado tempo.

A *blacklist* em si e as diversas outras variáveis de controle são gerenciadas pela base de dados em *cloud* [Redis 2012]. Esta base de dados é conhecida por sua simplicidade e eficiência. Ela basicamente mapeia **chave** e **dado**, oferecendo tempos de escrita e de leitura correspondentes à *hashing*. A implementação da *blacklist* foi feita utilizando o endereço IP de um cliente como chave, e o tempo que este cliente permanecerá bloqueado como dado. Por ser, indiretamente, um mecanismo de *hashing*, o tempo de busca por um cliente será O(1), o que é excelente para um mecanismo que filtrará todo tráfego que chega à aplicação.

5. Avaliação

A avaliação da arquitetura proposta consiste na análise da capacidade do servidor em atender novas requisições, sendo que o atendimento pode ser apenas o redirecionamento. Se o ataque de DDoS for devidamente mitigado, as requisições dos atacantes serão ignoradas, após a sua inclusão na *blacklist*. Logo, o servidor na *cloud* deverá ser capaz de redirecionar apenas clientes legítimos para a nova instância e garantir que eles terão acesso direto nas próximas requisições.

Para a experimentação, como nós atacantes, foram utilizadas oito máquinas do laboratório de pesquisa para processar os ataques, observando-se uma latência de rede no intervalo de $3ms$ a $7ms$. Cada uma destas máquinas operou com 25, 50, 75 ou 100 instâncias de um *script* atacante, que utiliza o comando *curl* para bombardear o servidor com requisições HTTP do tipo *GET*. Tais experimentos foram realizados diversas vezes, obtendo resultados de comportamento similar. Quanto à hospedagem, foi utilizado um

dyno para cada aplicação. Para a *cloud* Heroku, um *dyno* é uma instância de servidor virtual isolado com 512MBs de RAM e 4 cores Intel Xeon X5550 @ 2.67GHz.

5.1. Resultados

Para explorar e validar a nossa proposta, foram realizados experimentos de acordo com as métricas especificadas em [Bellaiche and Gregoire 2008]. Primeiro, foi avaliado o impacto da mitigação de DDoS em tempo de resposta e, em seguida, em relação à taxa de resposta e *overhead*. Como observado na Figura 4, com um intervalo de confiança de 95%, o uso da arquitetura proposta, gráfico à esquerda, reduziu o tempo de resposta às requisições legítimas em comparação ao gráfico à direita, que mostra o tempo gasto para atender o mesmo número de requisições originadas sem o nosso mecanismo. Tal comportamento ocorre porque com o uso da arquitetura, a *blacklist* impedirá uma aplicação de responder ao mesmo cliente múltiplas vezes, garantindo ainda assim que o cliente legítimo seja capaz de atingir a nova instância.

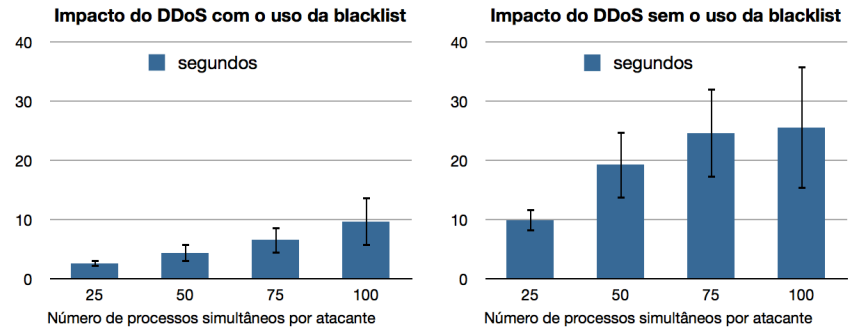

Figura 4. Tempo de resposta para clientes legítimos

Outra métrica avaliada é a taxa de páginas solicitadas recebidas com sucesso, mostrada na Figura 5. Há uma queda no número de respostas apenas quando não foram utilizadas a filtragem pela *blacklist* e o conseqüente redirecionamento. Nestes casos, a aplicação envia uma resposta ao atacante, que descarta esta resposta de imediato, dando continuidade ao ataque. Nota-se que a taxa de entrega sem o uso da *blacklist* é afetada pela ocorrência de *timeouts* de requisições HTTP não respondidas, pois a aplicação permanece ocupada com as requisições atacantes.

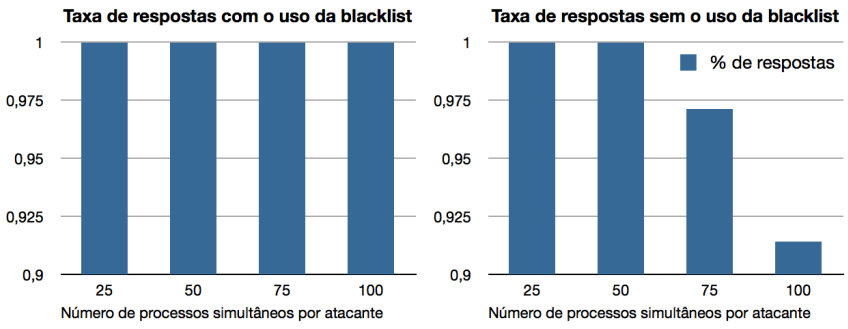

Figura 5. Taxa de respostas do servidor a clientes legítimos

6. Conclusão

Este trabalho apresentou uma arquitetura de mitigação para ataques de DDoS direcionados à aplicações web hospedadas em *cloud*. A arquitetura é dependente apenas da existência do ambiente na *cloud* e permite livre acesso aos clientes legítimos. O uso de uma *blacklist* é eficiente para filtragem devido ao tempo de validade dos registros, que no caso de atacantes, terá aumento exponencial para reincidências. Os resultados alcançados nas experimentações demonstram a validade da solução proposta, pois conseguem direcionar o tráfego legítimo de modo satisfatório, impossibilitando os atacantes de acessarem a nova instância criada. Mecanismos mais robustos para a checagem da *blacklist* em níveis mais baixos e otimizados serão desenvolvidos como trabalhos futuros, complementando a solução atual. Além disso, experimentos em maiores escalas tanto no ataque DDoS como também nos recursos alocados às instâncias da aplicação serão realizados.

Referências

Bakshi, A. and Yogesh1, B. (2010). Securing cloud from ddos attacks using intrusion detection system in virtual machine. In *Communication Software and Networks, 2010. ICCSN '10. Second International Conference on*, pages 260 – 264.

Bellaiche, M. and Gregoire, J.-C. (2008). Measuring defense systems against flooding attacks. In *Wireless Communications and Mobile Computing Conference, 2008. IWCMC '08. International*, pages 600 –605.

Dietrich, S., Goddard, N., and Long, N. (2000). Analyzing distributed denial of service tools: The shaft case. In *In Proceedings of USENIX LISA'2000*, pages 329–339.

Hazelhurst, S. (2008). Scientific computing using virtual high-performance computing: a case study using the amazon elastic computing cloud. In *Proceedings of the 2008 SAICSIT'08*, pages 94–103, New York. ACM.

Heroku (2012). http://www.heroku.com/.

Khor, S. H. and Nakao, A. (2009). spow: On-demand cloud-based eddos mitigation mechanism. *HotDep (Fifth Workshop on Hot Topics in System Dependability)*.

Liu, H. (2010). A new form of dos attack in a cloud and its avoidance mechanism. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, CCSW '10, pages 65–76, New York.

Peng, T., Leckie, C., and Ramamohanarao, K. (2007). Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.*, 39.

Pianese, F., Bosch, P., Duminuco, A., Janssens, N., Stathopoulos, T., and Steiner, M. (2010). Toward a cloud operating system. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 335 –342.

Redis (2012). http://redis.io/.

RubyOnRails (2012). http://rubyonrails.org/.

Sachdeva, M., Singh, G., Kumar, K., and Singh, K. (2008). Ddos incidents and their impact: A review.

Stavrou, A., Cook, D. L., Morein, W. G., Keromytis, A. D., Misra, V., and Rubenstein, D. (2005). Websos: an overlay-based system for protecting web servers from denial of service attacks. *Computer Networks*, 48:781–807.

Verkaik, P., Spatscheck, O., Van der Merwe, J., and Snoeren, A. C. (2006). Primed: community-of-interest-based ddos mitigation. In *Proceedings of the 2006 SIGCOMM*, LSAD '06, pages 147–154, New York. ACM.

