Ambientes Controlados de Geração de Anomalias: Uma Reprodução de Ataques de Negação de Serviço

Willian Alves Pereira Sukeyosi[1], Ekler Paulino de Mattos[2], Bruno Bogaz Zarpelão[3], Leonardo de Souza Mendes[2]

[1]Câmpus Coxim/Universidade Federal de Mato Grosso do Sul (UFMS)

[2]Faculdade de Engenharia Elétrica e de Computação (FEEC)/Universidade Estadual de Campinas (UNICAMP)

[3]Departamento de Computação/Universidade Estadual de Londrina (UEL)

sukeyosi@gmail.com, {emattos,lmendes}@decom.fee.unicamp.br, brunozarpelao@uel.br

Abstract: *In the last 30 years, communication networks have grown in size and complexity. As a consequence, the need for efficient network management systems also increased. However, network management researchers have faced difficulties to obtain data that can be used to evaluate their proposals. The objective of this work is to propose the building a testbed of denial of service attacks (Denial of Service, DoS) and present variables that characterize these attacks. The outcomes of this research were the construction of the testbed and the execution of DoS attack simmulations against the Web server Apache Tomcat 7.0.25.*

Resumo: *Nestes últimos 30 anos as redes de comunicação vem crescendo em tamanho e complexidade. Como consequência, cresceu também a necessidade de sistemas de gerência eficientes. Entretanto, os pesquisadores que trabalham com o desenvolvimento de métodos de gerência de redes tem encontrado dificuldade em obter dados para utilizar na avaliação de suas propostas. O objetivo deste trabalho é propor a construção de um ambiente controlado para gerar ataques de negação de serviço (Denial of Service, DoS) e apresentar variáveis que caracterizam estes ataques. Como resultados de nossa pesquisa, tivemos a construção do ambiente controlado e a realização de simulações de ataques de negação de serviço contra o servidor Web Apache Tomcat 7.0.25.*

1. Introdução

A crescente complexidade das redes de comunicação, com milhões de elementos operando em ambientes altamente dinâmicos, tem trazido novos desafios para a área de gerência de redes. Serviços novos e mais complexos como *peer-to-peer* (P2P), redes sociais, telefonia IP e transmissão de vídeos de alta definição geram grandes volumes de tráfego de dados nas redes de comunicação, consumindo grande quantidade de recursos da rede. Devido a este fato, a rede está sujeita a vários problemas, como a sobrecarga de recursos e falhas em equipamentos, que podem produzir comportamentos anômalos (ou anomalias).

Anomalias em redes de comunicação são eventos inesperados que causam um desvio acentuado do comportamento normal do tráfego de rede [Chandola et al., 2009]. Geralmente, as anomalias são causadas por defeitos em equipamentos, ataques, utilização abusiva da rede e falhas em softwares entre outros [Zarpelão 2010].Para evitar a degradação da qualidade dos serviços, devemos desenvolver sistemas de gerência de redes que sejam capazes de detectar as anomalias de maneira pró-ativa. Desta forma, os gestores da rede podem executar ações corretivas rapidamente, evitando que a prestação do serviço aos usuários seja prejudicada. Contudo, produzir sistemas de gerência de redes capazes de identificar com eficiência um comportamento anômalo é um desafio para os pesquisadores e para a indústria [Álvarez 2011]. Entre os desafios relacionados a esta tarefa, temos a dificuldade em construir ambientes para avaliar o desempenho dos sistemas de gerência de redes que trabalham com detecção de anomalias.

O objetivo deste trabalho é apresentar um ambiente controlado de geração de anomalias. Especificamente, vamos apresentar um ambiente controlado para gerar um tipo de anomalia conhecido como negação de serviço (*Denial of Service*, DoS). A negação de serviço é um tipo de ataque ocasionado por um agente malicioso, que dispara requisições contra um serviço até sobrecarregá-lo, provocando a sua negação para usuários legítimos. Serão apresentados resultados de simulações de dois cenários de ambientes controlados de ataque DoS contra um servidor Apache Tomcat 7.0.25, que geraram resultados satisfatórios.

O restante deste artigo está organizado conforme segue: 2. Ambientes controlados (*Testbed*s), 3. O ambiente controlado de geração de DoS – DoS *Testbed*, 4. Resultados, 5. Conclusão e 6. Referências Bibliográficas.

2. Ambientes Controlados (*Testbeds*)

Ambientes controlados de rede (*network testbeds*) são soluções que servem para simular ambientes reais. Eles são fundamentais para testar protocolos, serviços e sistemas em ambientes mais próximos da realidade com maior rapidez e eficácia.

Segundo Crepaldi et al. (2007), os *testbeds* dedicados a rede possuem os seguintes objetivos:

- Melhorar a produtividade na pesquisa e construção de sistemas complexos, como sistemas de gerência de redes, por exemplo.
- Devem ser fáceis de serem mantidos e devem exigir o mínimo de curva de aprendizagem possível do usuário. Se o *testbed* for difícil de configurar e entender, poderá comprometer o tempo de pesquisa.
- Devem possuir uma diversidade de protocolos de experimentos e cenários, fáceis de serem configurados para suportar várias topologias de redes com o mínimo de reconfiguração.
- Embora de tamanho minimizado, os *testbeds* devem parecer o mais próximo possível de um ambiente real. É necessário permitir o monitoramento e controle em tempo real dos protocolos e processos rodados em um *testbed*.

2.1.Trabalhos Relacionados

Xiaoyong et al. (2003) abordam ataques de negação de qualidade de serviço (*Denial-of-QoS*) em redes baseadas em serviços diferenciados (DiffServ). Este trabalho apresenta a

construção de um sistema de detecção de ataques de QoS baseados nas técnicas de detecção de anomalias: Qualidade do ajuste X^2 (X^2 *Goodness-of-fit*) e EWMA (*Exponential Weighted Moving Average*). Uma das contribuições deste trabalho foi a construção de um *testbed* para a reprodução dos ataques de negação de QoS.

Crepaldi et al. (2007) abordam um *testbed* chamado SignetLab para redes de sensores. Esta ferramenta funciona como um sistema de gestão de *testbed*: permite aos usuários programar, interagir e receber dados a partir de nós da rede. Enfatizam também que um *testbed* deve ser útil, e não deve sobrecarregar os pesquisadores com tarefas de manutenção e elevadas curvas de aprendizagem.
Heah e Zaidi (2009) apresentam resultados experimentais de um *testbed* para um esquema simples de recuperação de rotas locais para redes sem fio. O esquema é adaptativo. Quando um roteador é sobrecarregado com requisições, ou ocorre uma falha nele, o esquema procura uma nova rota isolando o nó sobrecarregado. Os ataques usados foram um tipo de ataque DoS conhecido como *Ping flooding* e falhas em nós da rede.

No trabalho de Hong et al. (2011), é proposto um *testbed* para sistemas de energia, automação de subestações e sistemas SCADA (*Supervisiory Control and Data Acquisition*). O sistema SCADA é um sistema de monitoramento em tempo real, de detecção de anomalias e análise de impactos para a rede de energia elétrica. O *testbed* oferece cenários para testar intrusões em subestações de energia elétrica.

Estes trabalhos propõem *testbeds* para testar sistemas de detecção de intrusão. De modo geral, não utilizam ferramentas conhecidas e também não oferecem detalhes de construção tornando-se difícil a composição destes *testbeds*. Além disso, o manuseio destas ferramentas e manutenção do próprio ambiente requer considerada curva de aprendizagem para os pesquisadores.

O trabalho proposto é a produção simplificada de um *testbed* que possibilita produzir cenários de ataques de maneira eficiente, conforme descrito na seção seguinte.

3. O ambiente controlado de geração de DoS – DoS *Testbed*

Esta seção apresenta o ambiente controlado de geração de ataques de negação de serviço (DoS *testbed*). São detalhadas as variáveis que caracterizam estes ataques, a estrutura física e os sistemas utilizados para a sua concepção.

3.1 Ataques de negação de serviço e sua caracterização

Os ataques de negação de serviço (*Denial-of-Service*, DoS) são caracterizados por uma tentativa explícita de impedir que um usuário legítimo utilize um determinado serviço. Neste tipo de ataque, um agente malicioso envia uma grande quantidade de requisições ao servidor de forma que recursos de memória e processamento sejam consumidos até a interrupção do serviço [Zarpelão 2010].

O DoS é identificado pelo número de pacotes SYN que chegam em um destinatário, e não são respondidos por ele ou pelo número de pacotes FIN que não recebem a sua confirmação de pacotes ACK [James 2011]. Além disso, a entidade atacada consome mais recursos de CPU e memória do que o normal [Vellalacheruvu 2011].

Levando em consideração estes fatos, as variáveis que identificam um ataque DoS em uma entidade atacada são: (i) número de pacotes TCP SYN recebidos/segundo; (ii) número de pacotes TCP SYNACK enviados/segundo; (iii) número de pacotes TCP SYNACK recebidos/segundo; (iv) número de pacotes TCP FIN recebidos/segundo; (v) número de pacotes TCP FIN enviados/segundo; (vi) uso de Memória; (vii) carga de CPU.

3.2 Visão geral do DoS *Testbed*

O ambiente controlado de geração de DoS é composto pelos seguintes elementos, conforme apresentando pela Figura 1-a:

- **Alvo**: Processo ativo (um serviço) que receberá requisições de entidades atacantes.
- **Atacante**: Processo que dispara requisições contra o alvo. Pode existir mais de um no ambiente.
- **Monitor**: Processo que deve monitorar, em tempo real, falhas em serviços e variáveis que medem o desempenho da entidade alvo, como a carga de CPU e uso de memória, por exemplo.
- **Analisador de tráfego**: Responsável por monitorar o tráfego da rede ou de um nó. No contexto do ambiente controlado, o analisador de tráfego é necessário para analisar o tráfego da rede e detalhes dos pacotes TCP que chegam à entidade alvo.

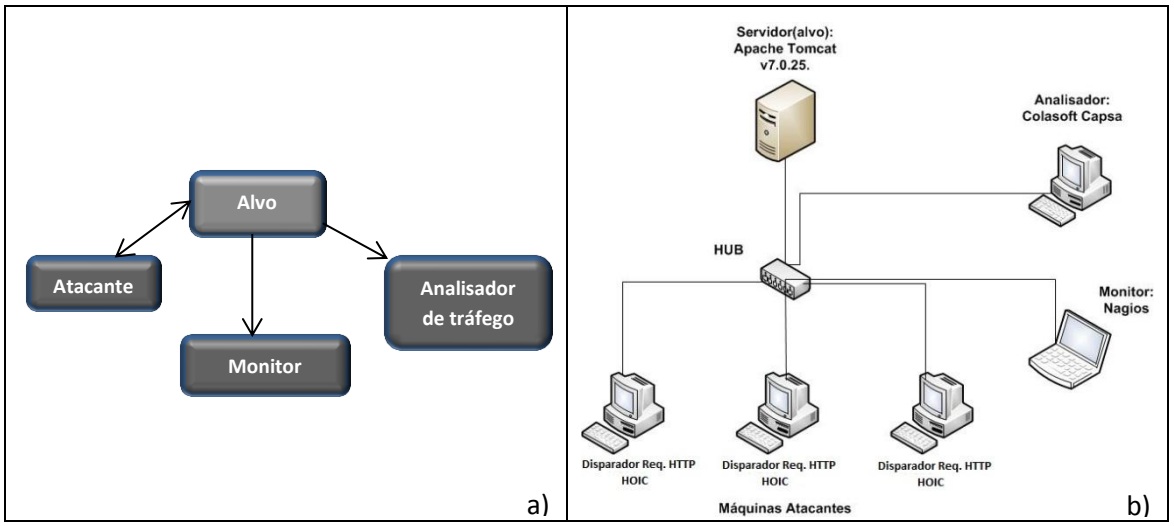

Figura 1. Visão do ambiente controlado: a) conceitual e b) física.

No DoS *testbed*, foram utilizadas entidades atacantes distribuídas em três máquinas. As entidades atacantes são representadas pelas ferramentas de ataque DoS: HOIC [Hoic 2012] e o Disparador de Requisições HTTP. O alvo é representado pelo servidor de aplicação Apache Tomcat versão 7.0.25. O monitor é o sistema de monitoramento Nagios, responsável por monitorar o status de execução e as variáveis de desempenho do alvo, como por exemplo, a carga de CPU e uso de memória. O analisador de tráfego é o sistema Capsa [Capsa 2012], que permite analisar a quantidade de pacotes TCP que chegam ao alvo e não são correspondidos. Em outras palavras, é responsável por verificar no alvo, a chegada de pacotes SYN e a sua resposta (pacotes SYNACK), e também analisar a presença de pacotes FIN enviados e recebidos. Esta

estrutura é apresentada pela Figura 1-b. Na seção seguinte, estas aplicações serão apresentadas com detalhes.

3.3 Software e Hardware

Nesta seção são apresentados os softwares utilizados para a produção do DoS *Testbed*.

HOIC - O HOIC é programa desenvolvido em C# utilizado para gerar ataques DoS e DDoS em redes de computadores com a finalidade de testar o desempenho de servidores. É possível configurá-lo para criar *threads* que disparam requisições HTTP simultaneamente para um servidor [Hoic 2012].

Disparador de Requisições HTTP – É um gerador de DoS, desenvolvido na linguagem JAVA pelo Laboratório de Suporte e Desenvolvimento (LSD) da Universidade Federal de Mato Grosso do Sul - UFMS do campus de Coxim – MS. Ele foi configurado para criar 100 *threads* que simultaneamente disparam 1000 requisições HTTP cada.

Colapsa - É um analisador de tráfego de rede para LAN e WLAN. Realiza a captura de pacotes em tempo real e faz monitoramento de rede 24/7. Possui um avançado analisador de pacotes e produz relatórios detalhados do comportamento da rede [Capsa 2012].

Nagios - É um software de monitoramento em tempo real de desempenho e de falhas em serviços e equipamentos de rede. Com o Nagios, é possível monitorar comutadores, roteadores, impressoras e serviços entre outros. Sua arquitetura permite adicionar plug-ins desenvolvidos nas linguagens Bash, C, Perl, Python, PHP, C# e CGI. É ainda dotado de uma interface Web que possibilita ter uma visão do status atual da rede, bem como, obter relatórios de utilização de recursos de um nó específico [Nagios 2012].

Na Tabela 1 é descrita a estrutura física utilizada no DoS *Testbed*.

Tabela 1. Descrição da estrutura física.

IP	Tipo	Configuração das Máquinas	Softwares instalados
10.0.0.1	Atacante	PC- Celeron CPU 2.53 GHz, 496 MB de RAM.	HOIC, LanTraffic V2, disparador de requisições HTTP.
10.0.0.2	Atacante	PC- Celeron CPU 2.53 GHz, 224 MB de RAM.	HOIC, LanTraffic V2, disparador de requisições HTTP.
10.0.0.10	Alvo	PC- Celeron E1400 CPU 2.00 GHz, 1,00 GB de RAM.	Servidor TomCat versão 7.0.25.
10.0.0.11	Atacante	PC- Celeron CPU 2.53 GHz, 1,00 GB de RAM.	HOIC, LanTraffic V2, disparador de requisições HTTP.
10.0.0.26	Monitor	Notebook - Intel core I3 CPU M370, 2.40 GHz 4,00 GB de RAM	Nagios
10.0.0.3	Analisador de tráfego	PC- Celeron CPU 2.53 GHz, 1,00 GB de RAM.	Colasoft Colapsa

4. Resultados

Neste trabalho, foram produzidos dois cenários do DoS *Testbed*, que serviram para investigar se as variáveis de caracterização apresentadas refletiram a mudança no tráfego, mesmo com ferramentas de ataques diferentes; também serviram para verificar a eficiência em reproduzir um ataque de negação de serviço, contra o servidor de aplicação Apache Tomcat-7.0.25 e medir o comportamento deste.

4.1 Cenário 1: ataques DoS com HOIC

A primeira simulação durou em torno de oito minutos e meio. Utilizamos três entidades atacantes com o software HOIC disparando requisições contra uma entidade alvo, a página inicial do servidor de Apache Tomcat-7.0.25. O HOIC foi configurado para criar 35 threads enviando requisições HTTP contra a porta 80 do servidor.

Planejamos o ataque de forma que o servidor Tomcat ficasse pouco tempo fora do ar em vários momentos da simulação. O objetivo foi testar o controle da geração do DoS e analisar se suas variáveis refletem a mudança no tráfego. Utilizamos o analisador de pacotes Capsa para monitorar o tráfego TCP no servidor.

A Figura 2 apresenta o tráfego recebido, em pacotes, pelo servidor atacado. Nota-se que houve um aumento no tráfego no tempo 16:51:21, quando a primeira entidade atacante foi disparada. É possível analisar o aumento gradativo de pacotes SYN Received, nos tempos 16:51:31, 16:52:01 e 16:53:06, chegando a atingir 6000 pacotes. Houve uma queda na resposta do servidor, em que o número de pacotes SYNACK Sent chega à zero. Este é um indício de que o ataque DoS está tendo sucesso.

Por volta do tempo 16:55:26, interrompemos o ataque de uma das instâncias do HOIC e o servidor voltou a atender às requisições. Neste tempo, os números de pacotes SYN Sent e SYNACK Sent se aproximam de 2400.

A partir do tempo 16:55:36 ativamos a instância do HOIC novamente, e a qualidade do atendimento do servidor Tomcat passou a se degradar até ficar inoperável. O número de pacotes SYN Received ultrapassou 6000, enquanto que o número de pacotes SYNACK Sent chegou a zero, no tempo 16:57:26.

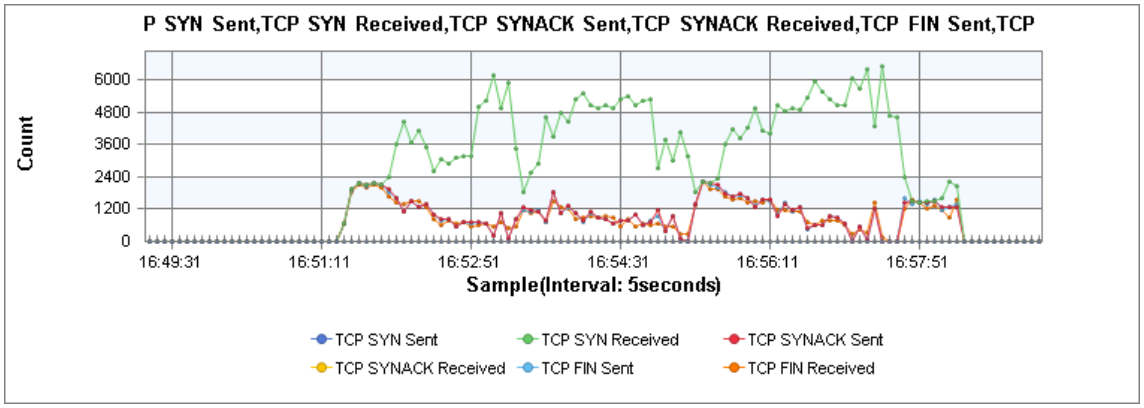

Figura 2 – Tráfego de pacotes na entidade alvo. Ataque gerado pelo HOIC

4.2. Cenário 2: ataques DoS com disparador de requisições HTTP

O disparador de requisições HTTP gerou requisições de 24576 bytes, por seis entidades atacantes distribuídas em três máquinas, que foram acionadas contra o servidor Tomcat. O ataque DoS durou em torno de 16 minutos. Planejamos o ataque de forma que o servidor Tomcat ficasse continuamente por mais tempo fora do ar do que a simulação anterior.

No tempo 17:11:01 todas as entidades atacantes já haviam sido disparadas, resultando num tráfego de 1500 pacotes (Figura 3). Neste tempo, os pacotes foram recebidos e respondidos pela entidade alvo, indicando o comportamento normal no tráfego.

A anomalia no servidor Tomcat, surge por volta do tempo 17:16:26. No qual, o número de pacotes SYN Received, recebidos das entidades atacantes, atinge cerca de 2250. Em compensação o servidor respondeu com menos de 1000 pacotes FIN Sent devido sobrecarga.

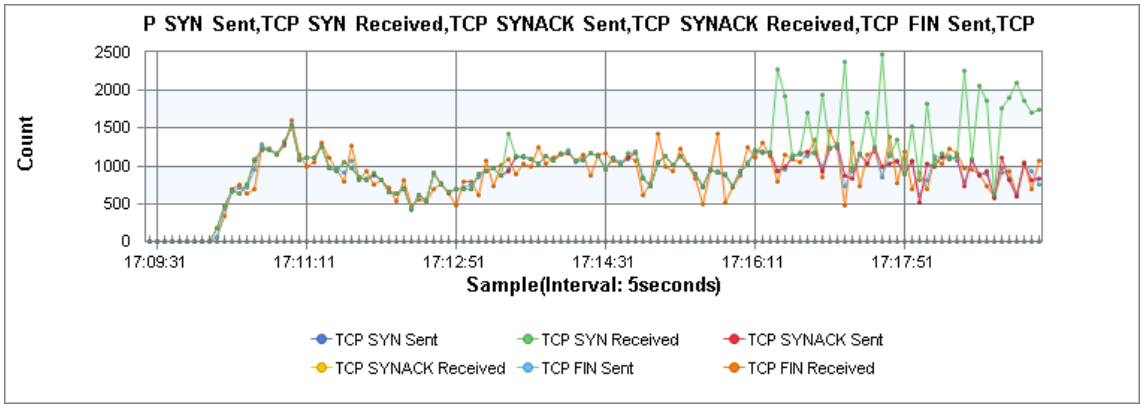

Figura 3 – Tráfego de pacotes na entidade alvo no período de 17:09:31 a 17:17:51. Ataque gerado pelo disparador de requisições HTTP.

A partir do tempo 17:18:51, a degradação da qualidade de atendimento do servidor passa a ser mais acentuada (Figura 4). No tempo 17:19:11 os valores de SYN Received e SYNACK Sent chegam a 2000 e 600 pacotes respectivamente. A quantidade de pacotes SYN Received e SYNACK Sent ficou bastante diferente, indício de que o servidor está sobrecarregado de requisições.

Por fim, no tempo 17:22:41, o número de pacotes SYN Received atingiu mais de 3000 pacotes sem resposta do alvo. A partir deste momento, o servidor Apache Tomcat ficou permanentemente inoperável.

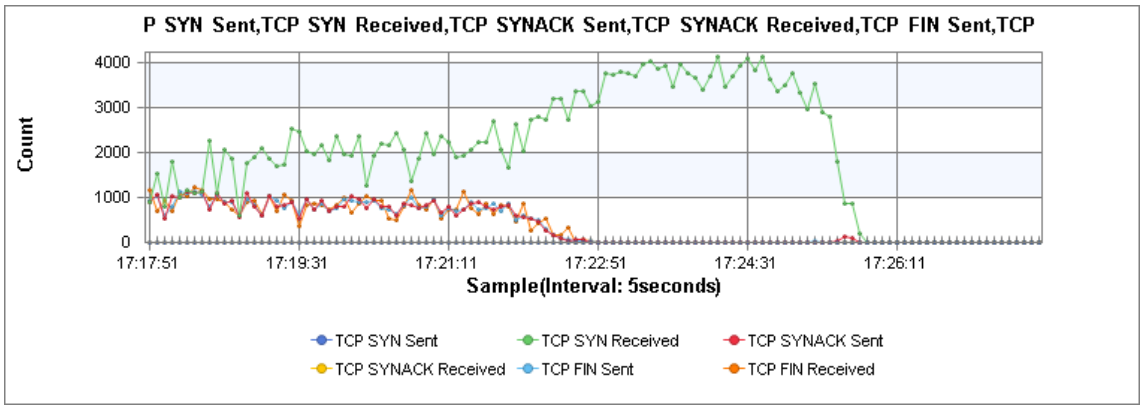

Figura 4 – Tráfego de pacotes na entidade alvo no período de 17:17:51 a 17:26:11. Ataque gerado pelo disparador de requisições HTTP.

4.3 Resultados: Monitoramento com o Nagios

As Figuras 5 e 6 apresentam o monitoramento com o Nagios das médias da carga de CPU e uso de memória, calculadas em intervalos de cinco minutos, do servidor Tomcat durante a execução dos cenários. Importante considerar que a máquina do Nagios apresenta seis minutos de atraso em relação à máquina do Capsa. O Nagios inicia o

monitoramento da carga de CPU e uso de memória em 16:45:00, enquanto que o Capsa registra as simulações em 16:51:00.

Na primeira simulação, com o HOIC, a média da carga da CPU chegou a 30% e o uso de memória a 550Mb (53% do total), no período das 16:45:00 até 16:55:00 (Figuras 5 e 6). A média da carga de CPU não atingiu um estado crítico, pois o Tomcat ficou fora do ar em torno de apenas setenta segundos, sendo que é neste momento em que ocorre maior consumo dos recursos da máquina. Este comportamento pode ser observado na Figura 2, em que temos as maiores diferenças entre a quantidade de pacotes SYN e SYNACK.

Na simulação com o disparador de requisições HTTP, os recursos da máquina foram mais explorados e o Tomcat ficou fora do ar por mais tempo. A simulação ocorreu no intervalo entre 17:10:00 e 17:25:00. Houve variação da carga de CPU e utilização de memória em dois momentos. No primeiro momento, entre 17:05:00 a 17:15:00, a utilização da CPU chegou a 60% e a utilização de memória chegou a 700Mb (68% do total). O servidor Tomcat estava respondendo às requisições das entidades atacantes, mas apresentando sinais de degradação na qualidade de seu atendimento.

Já no segundo momento, entre 17:15:00 a 17:25:00, a carga de CPU e utilização de memória sobem para 66% e 770Mb (75% de uso) respectivamente. O servidor Tomcat ficou indisponível por três minutos, deixando de atender às requisições das entidades atacantes.

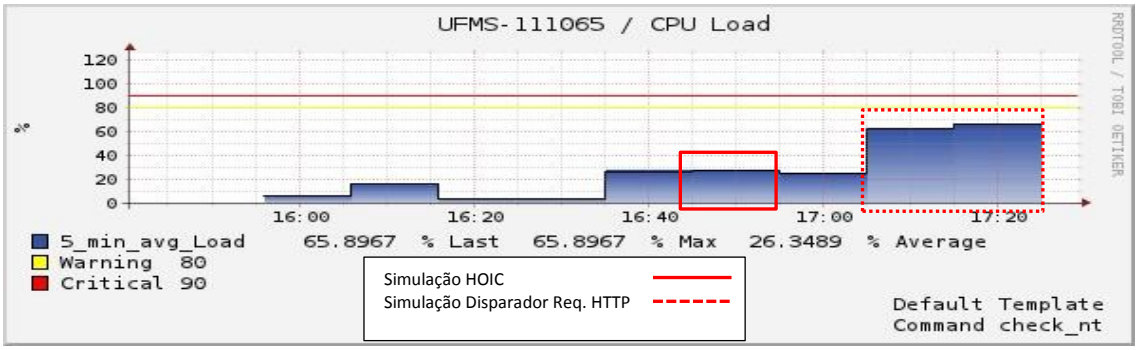

Figura 5 – Carga de CPU durante o ataque ao servidor Apache Tomcat (Monitoramento com o Nagios).

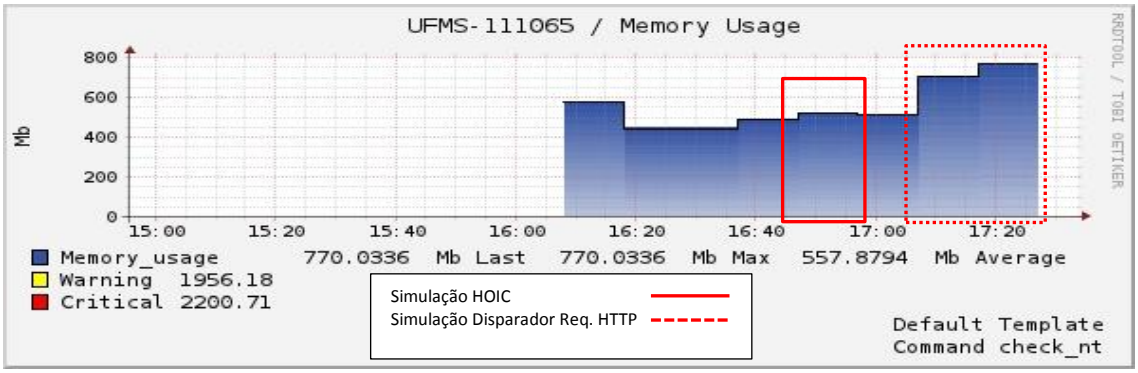

Figura 6 – Uso de memória durante o ataque ao servidor Apache Tomcat (Monitoramento com o Nagios).

Os resultados obtidos com o DoS *Testbed* foram satisfatórios. O DoS *Testbed* permitiu a geração de tráfego anormal e normal na rede, e a medição do

comportamento do servidor Tomcat nas duas situações. Isto foi possível com as variáveis de caracterização apresentadas, que acompanharam as mudanças no tráfego. Com o DoS *Testbed*, teremos subsídios para desenvolver e testar a eficácia de métodos para detectar anomalias.

5. Conclusão

Com base nos resultados obtidos nas simulações, conclui-se que tanto o disparador de requisições HTTP quanto o HOIC obtiveram êxito na reprodução do ataque DoS, já que eles conseguiram causar a interrupção das operações de um servidor Apache Tomcat-7.0.25.

Este trabalho abordou a construção de um ambiente controlado de geração de ataques de negação de serviço. Foram utilizadas ferramentas difundidas na área de redes, como o Nagios e o Capsa.

As contribuições desta pesquisa foram:

- Levantamento de variáveis que caracterizam um ataque, e ferramentas para construção de um ambiente controlado de geração de DoS.
- Desenvolvimento de uma ferramenta de geração de DoS.
- Construção de dois cenários de DoS com diferentes ferramentas de ataque.

A utilização do DoS *Testbed* permitiu produzir tráfego normal e anormal na rede de maneira controlada, e medir o comportamento de um servidor nas duas situações. As variáveis de caracterização de um ataque DoS propostas identificaram o momento em que uma anomalia ocorreu, e permitiram mensurar o impacto causado por ela no ambiente. Com o DoS *Testbed*, será possível desenvolver métodos para detectar anomalias e testar sua eficácia na ocorrência delas.

Este trabalho retrata apenas um dos módulos do ambiente controlado de geração de anomalias. Uma das propostas de trabalhos futuros é agregar outros tipos de anomalias no ambiente controlado, como por exemplo, anomalias geradas por agentes não maliciosos como *flashcrowd*, *Babbling node*, Tempestade de *broadcasts*, falhas em equipamentos de rede e de configuração.

Outra meta é produzir novos cenários para o DoS *Testbed* utilizando outros servidores de aplicação de código aberto. Entre eles podemos destacar GlassFish, JBoss, JOnAS e Apache Geronimo. Também é possível produzir melhores resultados de desempenho com o Apache Tomcat, ajustando seus parâmetros de configuração *KeepAliveTimeout* e *MaxClients*. Algumas propostas de configuração desses parâmetros podem ser encontrados em [Diao et al. 2003] e [Sugiki 2007].

6. Referências Bibliográficas

ÁLVAREZ, A. GARCÍA, R., CABRERO, S., PAÑEDA,X., MELENDI, D., OREA, R. (2011) "In Pursuit of Massive Service Emulation: A Methodology for *Testbed* Building". Proceedings of the IEEE Communications Magazine, DOI 10.1109/MCOM.2011.6011748, September.

CHANDOLA, V., BANERJEE, A., and KUMAR V. (2009) "Anomaly Detection: A Survey". Proceedings of the ACM Computing Surveys, v. 41, no. 3, p. 58, DOI 10.1145/1541880.1541882, July.

COLASOFT CAPSA (2012), disponível via WEB no endereço http://www.colasoft.com/capsa/, acessado em 20/10/2012.

CREPALDI, R., FRISO, S. ; HARRIS, A. ; MASTROGIOVANNI, M. ; PETRIOLI, C. ; ROSSI, M. ; ZANELLA, A. ; ZORZI, M. (2008) "The Design, Deployment, and Analysis of SignetLab: A Sensor Network *Testbed* and Interactive Management Tool". Proceedings of the IEEE TridentCom 2007. 3rd International Conference, p.1-10. DOI 0.1109/TRIDENTCOM.2007.4444656, February.

HEAH, D.; ZAIDI, Z.R. (2009) "A *Testbed* Implementation of Monitoring Assisted Local Route Recovery Scheme for Wireless Mesh Networks Advances in Mesh Networks". Proccedings of the IEEE - MESH 2009. Second International Conference. DOI: 10.1109/MESH.2009.29, p.120-125.

HOIC (2012), disponível via WEB no endereço http://www.eletrofox.com/loic/, acessado em 20/10/2012.

HONG, J., SHINN-SHYAN WU, STEFANOV, A. ; FSHOSHA, A. ; CHEN-CHING LIU ; GLADYSHEV, P. ; GOVINDARASU, M. (2011) "An Intrusion and Defense *Testbed* in a Power System Environment" proceedings of the IEEE Power and Energy Society General Meeting, p. 1- 5, July.

JAMES, C. (2011) "Time Series Models and its Relevance to Modeling TCP SYN Based DoS Attacks", proceedings of the IEEE Next Generation Internet (NGI), 2011 7th EURO-NGI Conference, p.1 – 8.

LANTRAFFIC (2012), disponível via WEB no endereço http://www.zti-telecom.com/EN/LanTrafficV2.html, acessado em 20/10/2012.

NAGIOS (2012), disponível via WEB no endereço http://www.nagios.org/, acessado em 20/10/2012.

VELLALACHERUVU, H. K, KUMAR, S. (2011) "Effectiveness of Built-in Security Protection of Microsoft's Windows Server 2003 against TCP SYN Based DDoS Attacks", proceedings of the Journal of Information Security, 2011, 2, 131-138 doi:10.4236/jis.2011.23013 Published Online July 2011 (http://www.SciRP.org/journal/jis).

LIU X .; SHA L.; YIXIN D.; FROEHLICH S.; HELLERSTEIN J. L.; PAREKH S. (2003), "On-line response time optimization of an apache web server," in International Workshop on Quality of Service (IWQoS), Monterey, CA, June 2003.

SUGIKI A.; KONO K.; IWASAKI H. (2007) "Tuning mechanisms for two major parameters of Apache web servers," Software Practice and Experience, vol. 38, 2008.

XIAOYONG, W.; MAHADIK, V.A.; REEVES, D.S. (2003) "A summary of detection of denial-of-QoS attacks on DiffServ networks" proceedings of the IEEE Conference - DARPA Information Survivability Conference and Exposition, v. 2, p. 277-282.

ZARPELÃO, B. B. (2010) "Detecção de Anomalias em Redes de Computadores", tese de doutorado, Universidade Estadual de Campinas FEEC/Unicamp, Campinas.