SIMULAÇÃO DE SYN FLOODING ATTACK NO COMMON OPEN RESEARCH EMULATOR

SYN FLOODING ATTACK SIMULATION IN COMMON OPEN RESEARCH EMULATOR

Alex M. S. Orozco*
Augusto P. Fernandes**
Giovani H. Costa***

* Professor do Instituto Federal Sul-rio-grandense – Campus Sapucaia do Sul/RS. Doutorando do Programa de Pós-graduação em Ciência da Computação da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS).
orozco@sapucaia.ifsul.edu.br

** Professor da Faculdade de Tecnologia Senac – Fatec – Porto Alegre/RS. Mestrando do Programa de Pós-graduação em Ciência da Computação da PUCRS.
augusto.fernandes@acad.pucrs.br

*** Professor do Centro Universitário Ritter dos Reis – Porto Alegre/RS. Mestrando do Programa de Pós-graduação em Ciência da Computação da PUCRS.
giovani.hoff@acad.pucrs.br

R e s u m o

Este trabalho aborda a utilização do Common Open Research Emulator (CORE) como plataforma para a simulação de ataques de negação de serviço (DoS - Denial of Service), como o ataque de inundação de SYN. O CORE permite facilmente projetar uma topologia de rede fictícia e definir serviços a serem executados nos componentes da rede. Com base na infraestrutura projetada, o ataque é disparado, e os dados que trafegam pela rede são analisados através de uma ferramenta de IDS/IPS (Intrusion Detection and Prevention System). Após a detecção do ataque, contramedidas são aplicadas buscando interromper o fluxo de dados entre o atacante e a vítima. De forma a efetivar as contramedidas, são utilizadas as ferramentas SNORT e Guardian. Esta estrutura permite que o processo de simulação ocorra antes de efetivar a aquisição da infraestrutura, diminuindo o índice de risco do projeto. Este ambiente permite também as atividades de ensino e treinamento na área de redes de computadores e segurança da informação e comunicação de forma simples.

P a l a v r a s - c h a v e : CORE. Ataque de negação de serviço. Sistema de prevenção e detecção de intrusões.

Abstract

This paper discusses the use of the Common Open Research Emulator (CORE) as a platform to simulate attacks of Denial of Service (DoS), as the SYN Flooding attack. The graphical interface of CORE easily allows to orchestrate a fictitious network topology and to define the services performed by the hosts. Through the structure designed, we can effectively attack a host and analyze the data flow by an IDS/IPS (Intrusion Prevention and Detection System). When the system detects the attack, the IPS applies countermeasures aiming to interrupt the data flow between the attacker and the victim. To perform the process of countermeasures, we use tools such as SNORT and Guardian. This structure allows the simulating process to occur before purchasing the equipment, reducing the risk level of the project. This environment also provides learning and training activities focused on computer network and information and communication security, and communication in a simple fashion.

Keywords: CORE. Attack of Denial of Service. Intrusion Prevention and Detection System.

1 Introdução

A constante evolução das tecnologias de informação e comunicação vem exigindo um crescente esforço da sociedade na tentativa de proteger as informações que circulam pelas mais diversas infraestruturas de rede. Em virtude dos mais diversos interesses envolvidos em acessar dados sensíveis ou evitar que os dados cheguem ao destinatário, as técnicas de ataques são elaboradas com um grau de refinamento cada vez maior. Para desenvolver a proteção contra essas diversas formas de ataques, as contramedidas necessitam ser experimentadas em ambientes complexos e heterogêneos, o que envolve um alto custo no processo.

Uma forma de proporcionar um ambiente de investigação com baixo custo ocorre através de ambientes que sejam capazes de emular redes compostas por diversos equipamentos, tais como roteadores e *hosts*, através de virtualização, e também sejam capazes de simular os meios de comunicação.

A ferramenta CORE (*Common Open Research Emulator)* proporciona essas funcionalidades (AHRENHOLZ *et al*., 2008). Esse tipo de ambiente pode fornecer uma estrutura para simular diferentes tipos de ataques, como os categorizados como Ataque de Negação de Serviço, do inglês *Denial of Service* (DoS), considerado o método de ataque mais comum realizado por invasores em uma rede, o qual pode apresentar efeitos catastróficos em áreas com recursos restritos (NEMADE *et al*., 2014).

O ataque DoS tem como objetivo negar o acesso a usuários em um servidor, através do envio de pacotes de dados em massa, visando a superar a capacidade de processamento do alvo, consumir os recursos do sistema ou de largura de banda, o que resulta na paralisia dos serviços de rede. Qualquer ação que pode impedir os usuários de usarem um serviço e de usufruírem do comportamento normal dos serviços de rede pode ser denominada como ataque de DoS (KAVISANKAR; CHELLAPPAN; VAISHNAVI, 2014).

Os dois protocolos mais populares usados na camada de transporte são o Protocolo de Controle de Transmissão, do inglês *Transmission Control Protocol* (TCP)[1] (DUKE *et al*., 2006), e o Protocolo de Datagrama do Usuário, do inglês *User Datagram Protocol* (UDP)[2] (POSTEL, 1980). Um dos principais riscos na segurança da camada de transporte, associado com TCP, é o denominado ataque de inundação de TCP SYN. Um ataque de inundação de SYN é uma forma de ataque de DoS em que o atacante envia uma sucessão de requisições SYN para o sistema alvo, com o objetivo de consumir os recursos existentes no servidor de forma a torná-lo indisponível para o tráfego legítimo (KAVISANKAR; CHELLAPPAN; VAISHNAVI., 2014).

O restante deste trabalho está organizado na seguinte estrutura: inicialmente, na Seção 2, apresentar-se o referencial teórico utilizado; na Seção 3, há o trabalho desenvolvido; os resultados obtidos são analisados na Seção 4, e a Seção 5 conclui o estudo realizado.

[1] Protocolo de comunicação da camada de transporte.

[2] Protocolo simples de comunicação da camada de transporte.

2 Referencial teórico

O objeto de estudo deste artigo necessita de uma série de conhecimentos intrínsecos ao seu funcionamento. Sendo assim, os conhecimentos teóricos envolvidos neste trabalho são discriminados a seguir.

2.1 Ataque de inundação de syn

Esta forma de ataque explora a fraqueza da especificação do protocolo TCP. No protocolo TCP, uma comunicação entre uma origem e um destino ocorre através de uma conexão estabilizada em um processo denominado *3-way handshake* (SCHUBA *et al*., 1997), ilustrado na Figura 1.

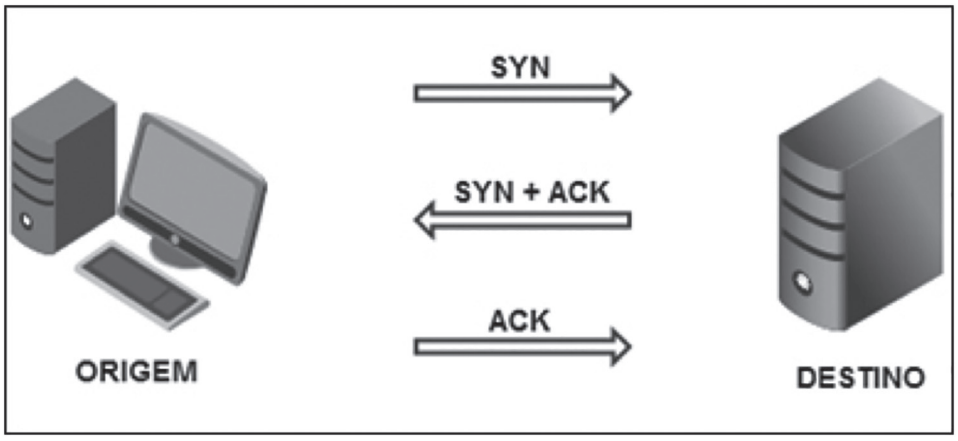

Figura 1: 3-way handshake
Fonte: Schuba et al., 1997

Inicialmente, a origem envia um pacote SYN (*SYNcronize*), com o seu número de sequência, para o destino. Ao receber este pacote, a conexão é considerada no estado de *parcialmente aberta*. A conexão estará neste estado até que o tempo limite de conexão seja atingido, normalmente 75 segundos (SCHUBA et al., 1997). O destino possui uma fila de conexões parcialmente abertas, de forma a gerenciá-las. O destino responde a essa solicitação SYN com um pacote ACK (*ACKnowledge*), confirmando o número de sequência da origem, e enviando também a sua solicitação SYN, com outro número de sequência, para a origem. Ao receber a resposta e a solicitação do destino, a origem também responde com um pacote ACK, confirmando o número de sequência do destino (POSTEL, 1981). Para realizar o ataque, são enviadas rajadas de requisições SYN fictícias, de forma a comprometer a capacidade de tráfego da rede, ou extrapolar a fila de conexões parcialmente abertas, impedindo que conexões verdadeiras sejam aceitas.

Uma técnica de detecção para o ataque de inundação de *SYN* pode se concentrar na carga útil e na área inutilizável do protocolo TCP. Esta técnica é focada no monitoramento do tráfego e filtragem dos pacotes. Assim, a verificação do desempenho da rede é um modo de detectar uma corrente anormal que pode

ser causada por inundação de *SYN*. Ainda, esta análise gera como resultados as diferenças entre o fluxo normal e o fluxo malicioso (HARIS et al., 2010).

2.2 *Common Open Research Emulator* (Core)

O emulador de redes CORE é uma ferramenta para emular redes de computadores em uma ou mais máquinas utilizando código aberto. O CORE consiste de uma interface gráfica para criação de topologias sobre máquinas virtuais leves e utiliza módulos Python para emular scripts de rede. O CORE foi desenvolvido por um grupo de pesquisa para tecnologias de rede, que faz parte da divisão de pesquisa e tecnologia da Boeing (NAVY, 2014). A marinha dos Estados Unidos oferece suporte no desenvolvimento deste projeto com código fonte aberto.

O CORE é especificamente usado para simulação de redes e protocolos de pesquisa, demonstrações, teste de aplicativos da plataforma, avaliar cenários de redes, realizar estudos de segurança e aumentar o tamanho das redes de teste físico. Ele fornece um ambiente para execução de aplicações e protocolos reais, aproveitando a virtualização fornecida pelo sistema operacional Linux[3].

[3] Sistema operacional, de código fonte aberto, criado em 1991, por Linus Torvalds, na universidade de Helsinki na Finlândia.

2.3 Sistemas de Detecção de Intrusão

A Detecção de Intrusão é uma das áreas de maior expansão, pesquisa e investimentos em segurança para redes de computadores. Com isso, são utilizados Sistemas de Detecção de Intrusão (*Intrusion Detection System* – IDS), que são ferramentas inteligentes capazes de detectar tentativas de invasão em tempo real. Esses sistemas podem atuar de forma a somente alertar as tentativas de invasão, como também em forma reativa, aplicando ações necessárias contra um ataque (SNORT TEAM, 2014). Um IDS pode ser classificado em dois tipos principais:

a) Sistemas Baseados em Rede (*Network Intrusion Detection System* - NIDS) – Estes tipos de aplicações são colocados na rede, perto do sistema ou dos sistemas a serem monitorados. Eles examinam o tráfego de rede e determinam se estes estão dentro de limites aceitáveis;
b) Sistemas Baseados em Host (*Host-Based Intrusion Detection System* - HIDS) – Estes tipos de aplicações rodam no sistema que está sendo monitorado e examinam o sistema para determinar quando a atividade no mesmo é aceitável.

O *Open Source Network Intrusion Detection System* (SNORT) é uma ferramenta NIDS de código aberto bastante popular por sua flexibilidade nas configurações de regras e constante atualização frente às novas ferramentas de

invasão. Outro ponto forte desta ferramenta é o fato de ter o maior cadastro de assinaturas, ser leve, pequena, fazer escaneamento do sistema e verificar anomalias dentro de toda a rede ao qual seu computador pertence (SNORT TEAM, 2014).

O Guardian (STEVENS, 2014) é uma ferramenta que atua em conjunto com o SNORT, atualizando automaticamente as regras de vários firewalls[4] com base em alertas gerados pelo SNORT. A interação entre o SNORT e o Guardian possibilita ações reativas em caso de intrusão e prevenção de ataques futuros.

[4] Dispositivo de uma rede de computadores que tem por objetivo aplicar uma política de segurança a um determinado ponto da rede.

3 Descrição do trabalho desenvolvido

Inicialmente, foi elaborada uma topologia de rede de forma a representar uma situação factível a execução do ataque.

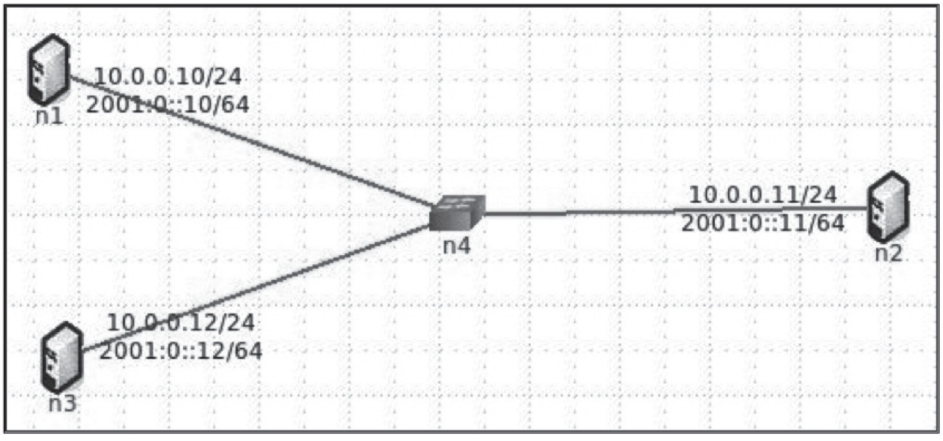

Figura 2: Topologia definida no CORE

Fonte: os autores

Conforme Figura 2, pode-se definir a topologia da seguinte forma:

a) **n1** – Ponto de partida do ataque de inundação de SYN, identificado pelo endereço IP 10.0.0.10;
b) **n2**: – Alvo do ataque de inundação de SYN, identificado pelo endereço IP 10.0.0.11;
c) **n3** – Computador virtual de acesso para testes funcionais da proposta, identificado pelo endereço IP 10.0.0.12;
d) **n4** – Switch para encaminhamento dos pacotes entre os segmentos da rede.

Para a execução do ataque, foi utilizada a biblioteca Scapy (GIFT; JONES, 2008), composta por um módulo da linguagem de programação Python que possui recursos para manipulação e geração de pacotes, monitoramento de rede, descoberta de rede, captura para análise de pacotes, entre outros. Desta forma, foi possível definir um conjunto de pacotes TCP para enviar requisições SYN malformadas até o alvo. O ataque envia pacotes TCP para a porta 80 da máquina alvo, a cada 0,3 segundos (PACHGHARE, 2011).

Em seguida, foram instanciadas na máquina alvo o servidor Web Apache (THE APACHE SOFTWARE FOUNDATION, 2014) e as ferramentas SNORT e Guardian, de modo que o SNORT detectasse o ataque e comunicasse ao Guardian a necessidade de aplicação de contramedidas para mitigar o ataque. Com o ataque bloqueado, o acesso ao servidor Web pela máquina **n3** deve estar em condições normais de utilização, já o fluxo entre **n1** e **n2** não deve apresentar tráfego.

Kuldeep e Tyagi (2014) assumem uma ideia parecida com o propósito deste trabalho, em que um atacante tenta inundar o alvo por inundação de SYN, mas usando mensagens ICMP. Ainda, eles utilizam como métrica um grande número de pacotes ICMP com tamanhos diferentes e nos quais é realizada a simulação do ataque através do simulador GNS3 (GNS3, 2014), que fornece uma interface gráfica de usuário para analisar redes complexas, parecido com o CORE. Porém, o GNS3 permite apenas a simulação de topologias de rede e o CORE permite também a emulação, permitindo conectar a topologia com redes reais.

4 Análise dos resultados obtidos

Inicialmente, com a execução do ataque, o tráfego entre os três computadores deve estar intenso, como é demonstrado nos gráficos da Figura 3. A linha grossa interconectando os equipamentos representa o alto tráfego de dados.

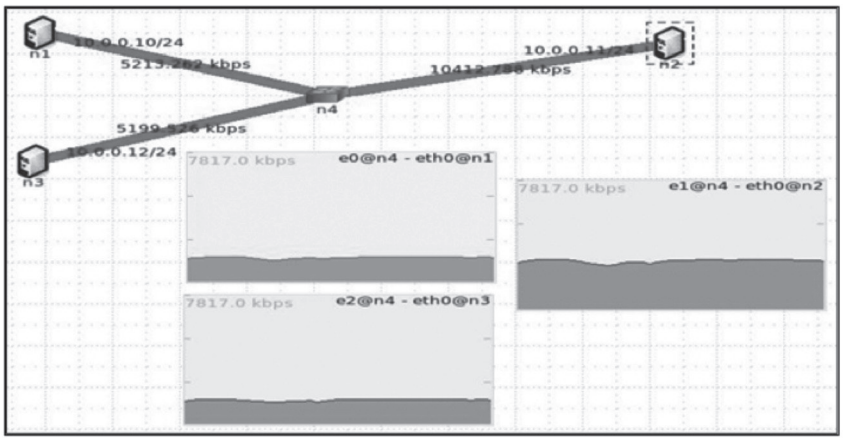

Figura 3: *Gráficos durante o ataque de SYN Flooding*

Fonte: os autores

Após a execução do SNORT e do Guardian, o ataque é detectado, e a contramedida é executada, cancelando a recepção de pacotes oriundos do endereço IP do atacante. A Figura 4 exibe a origem do ataque (10.0.0.10), o destino (10.0.0.11) e as portas encaminhadas (ssh e http). Quando o pacote está sendo aceito, a mensagem exibida aparece como RECV 2.

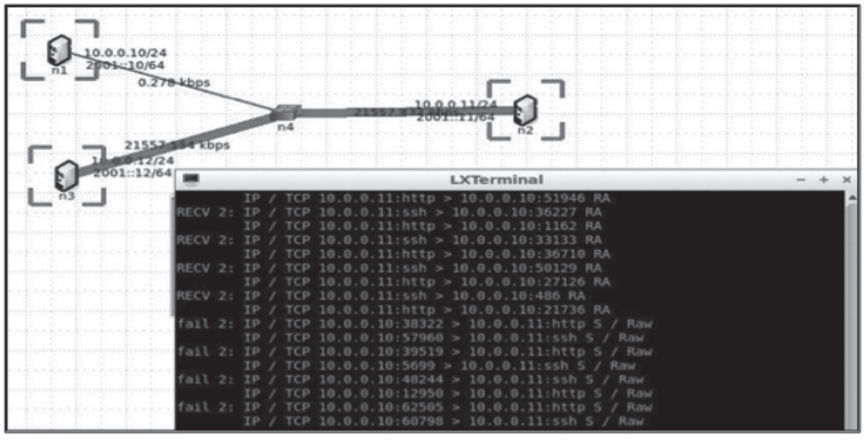

Figura 4: Visualização do ataque

Fonte: os autores

O bloqueio é realizado através da inserção de uma regra no *firewall*, inserida pelo Guardian, bloqueando todos os pacotes oriundos de 10.0.0.10, como ilustrado, através da mensagem *fail 2*, na Figura 5. A figura ilustra também a

obstrução do ataque pela linha fina entre **n1** e **n4**, representando o baixo tráfego oriundo da máquina atacante.

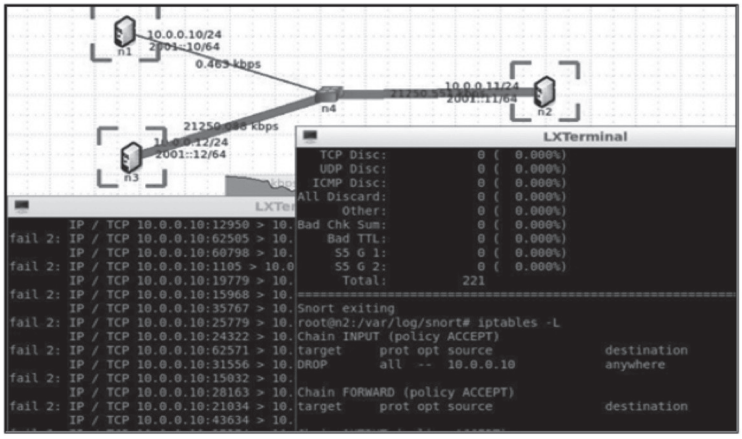

Figura 5: *Exemplo de bloqueio do ataque*

Fonte: os autores

Após a obstrução do ataque, o resultado da ação pode ser verificado nos gráficos da Figura 6, com uma diminuição brusca no tráfego entre **n4** e **n1** e um decréscimo perceptivo entre **n4** e **n2**.

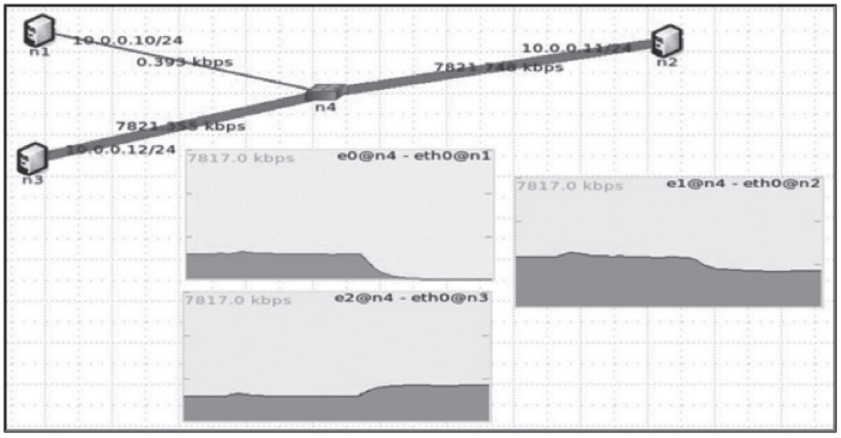

Figura 6: Gráficos após a obstrução do ataque

Fonte: os autores

5 Dificuldades encontradas

Houve dificuldade no uso do SNORT como IDS no ambiente do CORE. Cada vez que o CORE é iniciado, o SNORT não é inicializado corretamente, pois há a necessidade de criação do diretório /var/log/snort e do arquivo "alert" dentro deste diretório. Situações semelhantes foram identificadas com outros serviços como o HTTP (apache), por exemplo. Entretanto, essa situação pode ser sanada adicionando corretamente estes serviços na inicialização do nodo em teste através do CORE com a inclusão dos comandos necessários para a inicialização do serviço e com a correta declaração dos diretórios e arquivos requeridos pelo mesmo.

Outra anomalia no uso do SNORT refere-se ao registro de eventos no arquivo "alert". Em qualquer máquina da topologia, o SNORT não consegue escrever neste arquivo, funcionando de maneira correta somente na máquina principal. Para sanar este problema, o SNORT foi executado com o seguinte comando: "snort –A full –vde tcp –l /var/log/snort/", detectando todas as tentativas de ataque utilizando o protocolo "TCP" e gravando no arquivo "snort.log", localizado no diretório "/var/log/snort/".

Após a gravação do arquivo, para registro do evento foi necessário recuperar os dados contidos no mesmo e enviar ao arquivo "alert", com o comando "snort –r /var/log/snort/snort.log > /var/log/snort/alert". Esta ação é necessária, pois o Guardian monitora o arquivo "alert" para identificar ataques que estejam ocorrendo. Assim que o Guardian detecta alterações no arquivo "alert", executa as ações de contramedidas.

6 Conclusão

Após a execução do ataque do host **n1** contra o host **n2** e da execução de requisições do host **n3** contra o host **n2**, foi possível verificar que o Guardian somente toma contramedidas contra o host que fez o ataque de inundação de SYN, visto que a opção de detecção do SNORT continha o parâmetro "TCP". Assim que o registro de eventos do SNORT foi convertido para o "alert", o Guardian tomou uma ação criando uma regra no *firewall* e bloqueando o host **n1**, alcançando parcialmente o objetivo definido para este trabalho, em virtude da etapa manual de recuperação do alerta lançado pelo SNORT.

Com este cenário, foi verificado, no ambiente de teste, que o CORE pode apresentar complexidades de configuração que deveriam ser transparentes du-

rante o processo. Tal situação foi detectada na escrita do arquivo "alert" pelo SNORT, ocorrendo de forma satisfatória somente na máquina real.

Como trabalho futuro, pretende-se automatizar o processo de detecção de escrita no arquivo "alert", além de incluir as ferramentas SNORT e Guardian na configuração padrão do CORE. De forma a complementar este trabalho, pretende-se realizar novas simulações de ataques de inundação SYN, através de cenários com topologias mais complexas, com ataques distribuídos.

7 Agradecimentos

Os autores agradecem ao Prof. Dr. Avelino Francisco Zorzo, pelas contribuições e revisão crítica do artigo, e ao Prof. Dr. Tiago Coelho Ferreto, pelo estímulo ao desenvolvimento deste trabalho.

Referências

AHRENHOLZ, J. et al. *Core*: a real-time network emulator. In: *Ieee Military Communications Conference*, 2008, p. 1 – 7.

DUKE, M. et al. *RFC 4614*: a Roadmap for Transmission Control Protocol (TCP) Specification Documents. 2006. Disponível em: <https://tools.ietf.org/html/rfc4614>. Acesso em: Abr. 2014.

GIFT, N.; JONES, J. *Python for Unix and Linux System Administration*. Germany: O'Reilly, 2008.

HARIS, S. et al. *TCP SYN*: flood detection based on Payload Analysis. In: *Ieee Student Conference On Research And Development,* 2010, p. 149 – 153.

KAVISANKAR, L.; CHELLAPPAN, C.; VAISHNAVI, R. Network Layer DDoS Mitigation Model Using Hidden Semi-Markov Model. *International Journal of e-Education, e-Business, e-Management and e-Learning*, Califórnia, v. 4, n. 1, 2014, p. 42 – 46.

NAVY, U.S. Naval Research Lab. *Common Open Research Emulator (CORE)*, disponível em: <http://www.nrl.navy.mil/itd/ncs/products/core>. Acesso em: Abr. 2014.

NEMADE, S. et al. Early detection of SYN flooding attack by adaptive thresholding (edsat): a novel method for detecting SYN flooding based dos attack in mobile ad hoc network. *International Journal Of Advanced Research In Engineering And Technology*, Califórnia, v. 5, n. 2, 2014, p. 79 – 86.

PACHGHARE, S. *SYN Flooding using SCAPY and Prevention using ipta*bles. 2011. Disponível em: <http://www.opensourceforu.com/2011/10/syn-flooding-using-scapy-and-prevention-using-iptables>. Acesso em: Abr. 2014.

POSTEL, J. *RFC 768*: user datagram protocol. 1980. Disponível em: <https://tools.ietf.org/html/rfc768>. Acesso em: Abril de 2014.

POSTEL, J. *RFC 793*: transmission control protocol. 1981. Disponível em: <https://tools.ietf.org/ html/rfc793>. Acesso em: Abril de 2014.

SCHUBA, C. et al. Analysis of a Denial of Service Attack on TCP. In: *Ieee Symposium On Security And Privacy*, 1997, p. 208 – 223.

SNORT TEAM. *SNORT Users Manual*. Disponível em: <http://manual.snort.org>, Acesso em: Abril de 2014.

STEVENS, A. *Guardian Active Response for Snort.* 2002. Disponível em: <http://www.chaotic.org/guardian>. Acesso em: Abr. 2014.

THE APACHE SOFTWARE FOUNDATION. *Apache HTTP Server*. Disponível em: <http://httpd.apache.org>. Acesso em: Abr. 2014.

KULDEEP, T., TYAGI S. Enhancing network security by implementing preventive mechanism using GNS3. In: *International Conference On Reliability, Optimization And Information Technology*, 2014, p. 300 – 305.

GNS3. 2014. Disponível em: < http://www.gns3.net>. Acesso em: Abr. 2014.