1.03.03 - Metodologia e Técnicas da Computação

# ESTUDO E ANÁLISE DE TRÁFEGO DE REDES USANDO ALGORITMO DE INUNDAÇÃO

Igor A. C. Souza[1], Rubens B. Filho[2]
1. Discente do curso de Ciência da Computação da Universidade Estadual do Mato Grosso do Sul (UEMS)
2. Docente do curso de Ciência da Computação da Universidade Estadual do Mato Grosso do Sul (UEMS)

## Resumo

O ataque de negação de serviço (DoS – Denial of Service) consiste em impedir ou dificultar o acesso de usuários a serviços, tanto por consumo de processamento ou de memória do sistema quanto por transmissão de pacotes. Os alvos mais comuns dos ataques de negação de serviço são servidores web vulneráveis. Esse método não é realizado via invasão do sistema, mas invalida o alvo por sobrecarga. O propósito e motivação deste projeto é a implementação de um conjunto de algoritmos de inundação de rede, aplicando técnicas de inundação ICMP, UDP e TCP SYN. A análise dos resultados no quesito desempenho obtidos por meio da implementação algorítmica utilizando o modo DoS permitiu um aprofundamento dos conhecimentos necessários para evitar prejuízos financeiros e estruturais em servidores Web pertencentes a instituições e empresas privadas. Este projeto foi aplicado em um ambiente de teste real de pequeno porte (rede local) de uma Instituição de Ensino Superior. Os resultados obtidos com a validação da proposta foram comparados com resultados presentes na literatura da área. Buscou-se entender tanto as semelhanças quanto as diferenças entre os algoritmos implementados.

**Palavras-chave:** ataque de inundação, DDoS, redes de computadores.

## Introdução

Ataques de negação de serviços (DoS – Denial of Service) existem desde o surgimento das redes de computadores, mas não recebiam muita atenção até atingirem provedores de serviços de internet e grandes empresas. Este tipo de ataque tem por objetivo interromper ou prejudicar a disponibilidade de um serviço, impedindo o acesso ao mesmo (GOMES; ARAUJO; CAMPOS. 2015).

Ataques de negação de serviço distribuído (DDoS – Distributed Denial of Service) é uma variação do ataque de DoS que faz uso de uma rede de máquinas escravas para realizar um ataque sincronizado a um mesmo alvo. A utilização de diversas máquinas contribuindo para a dimensão do ataque, permite que sistemas maiores sejam afetados e dificulta a capacidade de se defenderem (GOMES; ARAUJO; CAMPOS. 2015).

Hackers vêm executando ataques de negação de serviço distribuído por décadas, e seu potencial vêm crescendo constantemente com o tempo (STALLINGS; BROWN, 2015). Devido ao crescimento da banda larga de internet, os maiores ataques têm crescido de 400 Mbps em 2002, para 100 gigabytes por segundo em 2010 e 300 Gbps no ataque a Spamhaus em 2013 (STALLINGS; BROWN, 2015). Atualmente, o maior ataque já registrado foi a plataforma de hospedagem de códigofonte GitHub, em Março de 2018. O site teve tráfego de 1.35 Tbps e ficou instável por, aproximadamente, 10 minutos (AUTRAN. 2018).

O problema abordado refere-se ao estudo do método de inundação utilizando diferentes protocolos. Ao aplicar o método de inundação supra cidade ocorre a troca de pacotes entre duas ou mais máquinas. Conforme abordado, a aplicação do método de inundação permitirá criar e identificar algoritmos eficientes para a realização de ataques e destacar o algoritmo que melhor desempenha a tarefa de envio de pacotes.

Uma vez que se trata de uma técnica que interfere na comunicação de aplicações, o uso dos métodos de inundação via pacotes dos tipos ICMP (Internet Control Message Protocol), UDP (User Datagram Protocol) e TCP SYN (Transmission Control Protocol Synchronization), podem gerar resultados satisfatórios na realização dos ataques de DoS.

O projeto tem como objetivo geral a elaboração de um estudo sobre tráfego de redes. Além disso, foram implementados um conjunto de algoritmos de inundação de rede, aplicando técnicas de inundação e analisado os resultados de cada algoritmo, avaliando seu desempenho.

## Metodologia

O projeto possui um desenvolvimetno progressivo linear, onde inicialmente, parte-se de uma revisão bibliográfica sobre assuntos gerais e específicos relacionados ao tema de estudo, como por exemplo o estudo sobre DoS e as diferentes técnicas de ataque. Livros referenciados na literatura que dão suporte a esse estudo são “Criptografia e Segurança de redes: Princípios e práticas (STALLINGS, 2008) ”, “Computer Security: Principles and practice (STALLINGS; BROWN, 2015) ” e “Network Security: a beginner’s guide (MAIWALD, 2013) ” foram importantes porque forneceram dados e conceitos sobre DoS, DDoS, as diferentes formas de ataques de inundação e as descrições dos protocolos utilizados. Além disso, foram utilizadas como referência, sites e artigos que também tiveram os livros citados anteriormente como base no assunto.

O desenvolvimento do projeto contou com a implementação dos algoritmos utilizando a linguagem de programação C. A cada implementação, os membros do projeto se reuniam para discutir os resultados parciais e, por meio de revisões do algoritmo almejar alterações pertinentes que poderiam melhorar a eficiência do

método.

Posteriormente, cada algoritmo passou por correção de erros e aprimoramentos, como, melhoramento de funções e reutilização de funções prontas tanto implementação pessoal quanto encontrados na internet. Cada aprimoramento realizado gerou diferentes versões que foram comparadas, umas com as outras, para adquirir resultados com relação a tempo de execução, eficiência e quantidade de pacotes enviados/processados por segundo. Todos os dados de performance dos algoritmos foram coletados de máquina de próprio uso particular. A máquina, da marca Dell, opera com o sistema operacional Kali Linux 2018.1, processador Intel Core i5-4600U 1.60GHz 2.30GHz.

Ao final da implementação dos métodos destacados, foi feita uma comparação utilizando variáveis de destaque que permitissem extrair uma melhor compreensão do estudo.

## Resultados e Discussão

Todos os resultados com os protocolos ICMP, UDP e TCP gerados durante o período de testes foram obtidos no analizador de tráfego de rede *Wireshark*. O *Wireshark* é um analisador de protocolo, que permite capturar e navegar interativamente no tráfego de uma rede de computadores em tempo de execução. O programa verifica os pacotes transmitidos pelos dispositivos de comunicação, como uma placa de rede (BRITO, 2014).

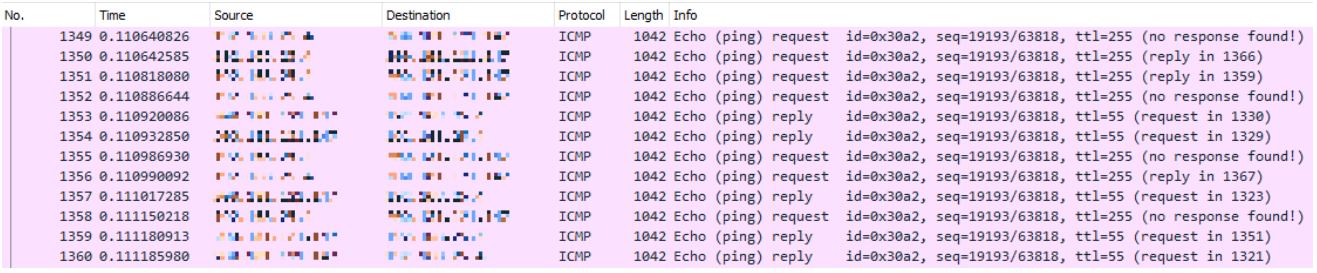
- Algoritmo A – inundação por ICMP

O algoritmo A utilizou o protocolo ICMP para realizar a tarefa de inundação. Além do *Wireshark*, no algoritmo A, também foi feita a utilização das funções *sigaction()* e *gettimeofday()* para identificar o tempo de execução e a quantidade de pacotes enviados, como uma maneira de obter uma noção inicial do funcionamento e desempenho do algoritmo. Nesse algoritmo foram criadas duas versões, uma utilizando a função *fast_rand()* e outra função *rand()*. A comparação dessas funções garantiu uma visão da eficiência de cada uma, dessa maneira, descobrindo qual função seria a mais adequada para ser utilizada nesse algoritmo. Com a utilização da função *fast_rand()*, apesar de terem sido gerados muitos pacotes durante os testes, foram coletados resultados ligeiramente mais eficientes de que a operação padrão *rand()*, como pode ser visto na Tabela 1.

| | Protocolo | Tamanho da carga (em bytes) | Pacotes enviados | Tempo de execução (em segundos) | Pacotes / Segundo |
|---|---|---|---|---|---|
| *Fast_rand()* | ICMP | 1000 | 1971111 | 180.222469 | 10937 |
| *Rand()* | ICMP | 1000 | 1923326 | 180.177694 | 10674 |

Tabela 1 – comparação do algoritmo de inundação ICMP utilizando *fast_rand()* e *rand()*

Os resultados com relação ao tempo de execução e a pacotes enviados e quantidade de pacotes por segundo, foram calculadas utilizando o analisador de tráfego de rede *Wireshark*. Antes de iniciar a execução dos algoritmos, o analisador foi ativado para fazer a captura dos pacotes enviados para um IP destino, como pode ser visto na figura abaixo. Os IPs de origem (source) e destino (destination) sendo utilizado, como na imagem abaixo, estão embaralhadas por não ser permitido exibi-las.



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1349 | 0.110640826 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) request id=0x30a2, seq=19193/63818, ttl=255 (no response found!) |
| 1350 | 0.110642585 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) request id=0x30a2, seq=19193/63818, ttl=255 (reply in 1366) |
| 1351 | 0.110818080 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) request id=0x30a2, seq=19193/63818, ttl=255 (reply in 1359) |
| 1352 | 0.110886644 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) request id=0x30a2, seq=19193/63818, ttl=255 (no response found!) |
| 1353 | 0.110920086 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) reply id=0x30a2, seq=19193/63818, ttl=55 (request in 1330) |
| 1354 | 0.110932850 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) reply id=0x30a2, seq=19193/63818, ttl=55 (request in 1329) |
| 1355 | 0.110986930 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) request id=0x30a2, seq=19193/63818, ttl=255 (no response found!) |
| 1356 | 0.110990092 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) request id=0x30a2, seq=19193/63818, ttl=255 (reply in 1367) |
| 1357 | 0.111017285 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) reply id=0x30a2, seq=19193/63818, ttl=55 (request in 1323) |
| 1358 | 0.111150218 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) request id=0x30a2, seq=19193/63818, ttl=255 (no response found!) |
| 1359 | 0.111180913 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) reply id=0x30a2, seq=19193/63818, ttl=55 (request in 1351) |
| 1360 | 0.111185980 | [redacted] | [redacted] | ICMP | 1042 | Echo (ping) reply id=0x30a2, seq=19193/63818, ttl=55 (request in 1321) |

Figura 1 – pacotes capturados no *Wireshark* no algoritmo A

Apesar da quantidade de pacotes enviados, o número de pacotes processados não foi muito grande, como pode ser visto a seguir:

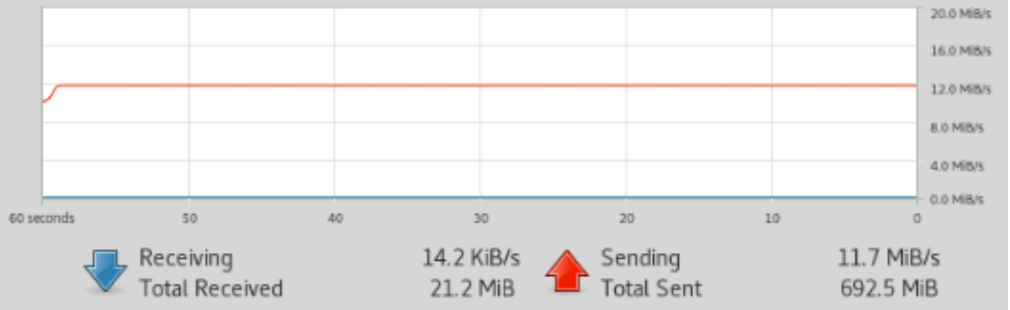



Figura 2 – gráfico da capacidade de envio de pacotes por segundo do algoritmo A

Como pode ser visto, a capacidade de envio de dados do algoritmo (linha vermelha), durante a execução, foi de aproximadamente 12.0 MiB /s, enquanto apenas 14.2KiB /s eram processados (linha azul).

- Algoritmo B – inundação por TCP SYN

Enquanto no algoritmo A foi realizada inundação por envio de pacotes *echo request*, o algoritmo B realiza inundação por envio de pacotes de requisição de conexão TCP. Os resultados do teste foram capturados no *Wireshark*. Os resultados adquiridos podem ser vistos na Tabela 2 a seguir.

| | Protocolo | Tamanho da carga (em bytes) | Pacotes enviados | Tempo de execução (em segundos) | Pacotes / segundo |
|---|---|---|---|---|---|
| Algoritmo B | TCP | 500 | 4679679 | 31.614147 | 148024 |
| Algoritmo B | TCP | 750 | 4656135 | 31.442411 | 148084 |
| Algoritmo B | TCP | 1000 | 4469497 | 38.215156 | 116956 |

Tabela 2 – resultados do algoritmo B

Este algoritmo foi criado de forma a alterar o IP após um interval de pacotes enviado, para evitar do IP ser bloqueado pelo servidor por envio excessive de pacotes. Entretanto, a quantidade de pacotes processados ainda não foi muito grande, como pode ser visto nas Fig. 3 e Fig. 4.

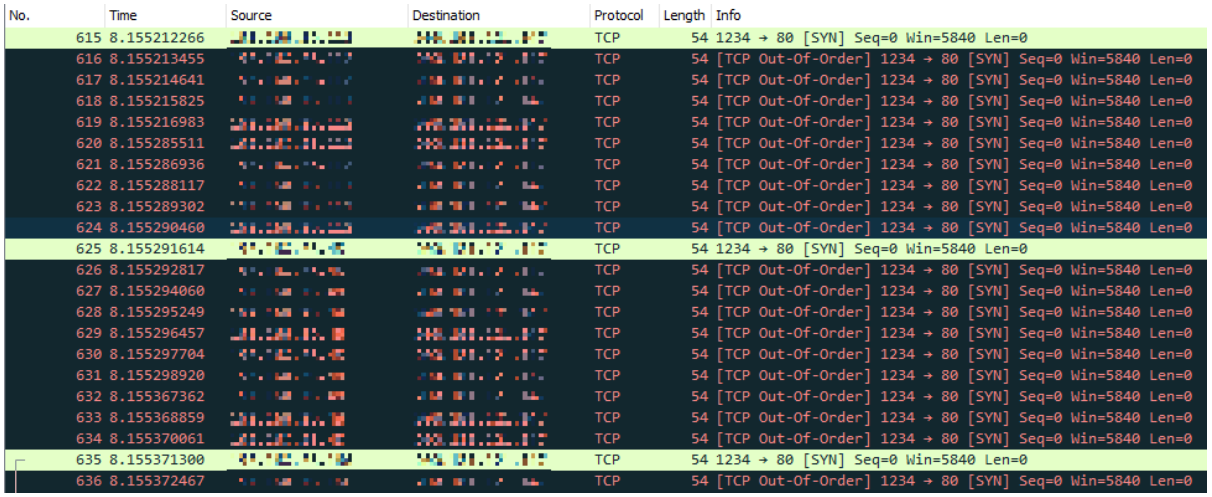



Figura 3 – pacotes TCP capturados pelo *Wireshark* no algoritmo B

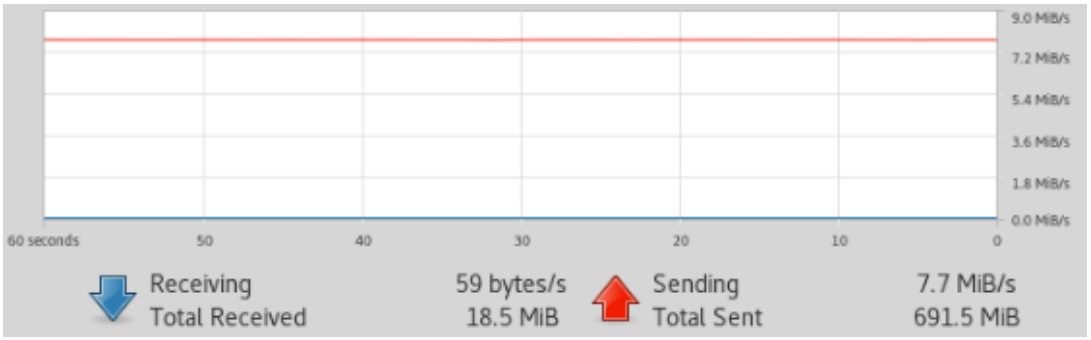



Figura 4 – gráfico com capacidade de envio de pacotes por segundo do algoritmo B

Como pode ser visto, a capacidade de envio de dados do algoritmo (linha vermelha), durante a execução, foi de aproximadamente 7.7MiB/s, enquanto, no momento, apenas 59bytes/s eram processados (linha azul).

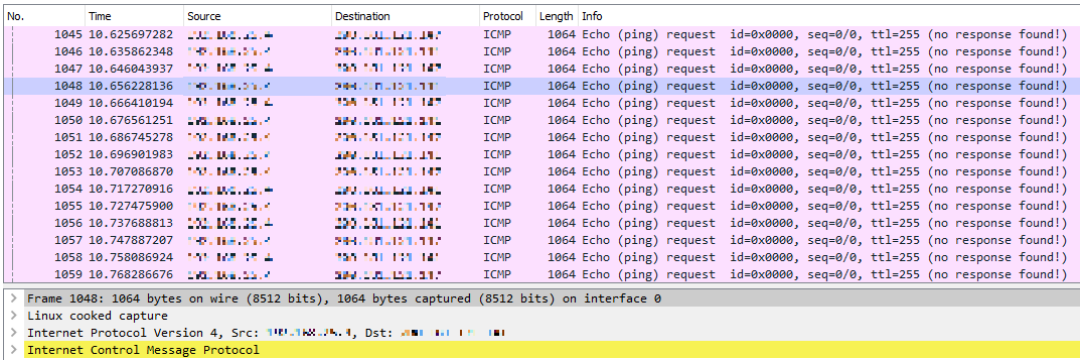
- Algoritmo C – inundação por ICMP/UDP

Neste terceiro algoritmo foi realizado um ataque de inundação tanto por pacotes *echo request* e UDP, os resultados do teste foram capturados no *Wireshark*. Os resultados adquiridos podem ser vistos na Tabela 3 a seguir.

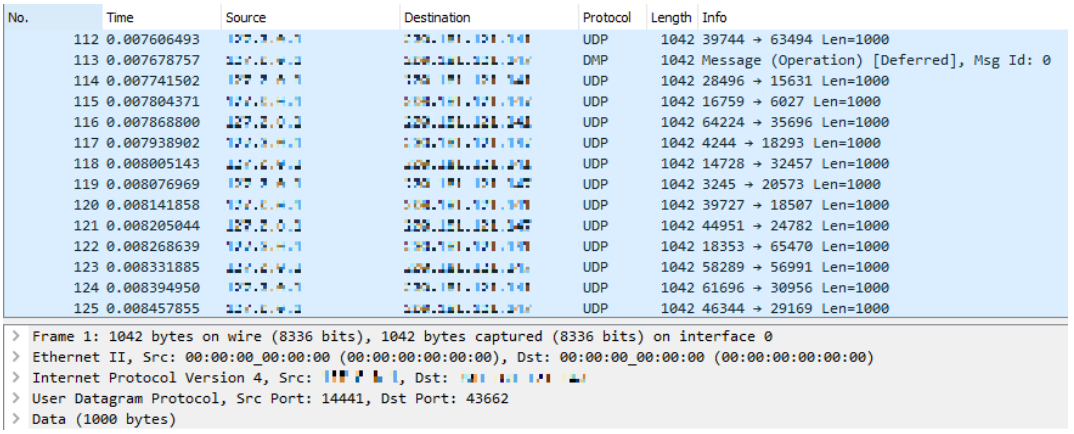
| | Protocolo | Tamanho da carga (em bytes) | Pacotes enviados | Tempo de execução (em segundos) | Pacotes / segundo |
|---|---|---|---|---|---|
| Algoritmo C | ICMP (padrão) | 500 | 1916743 | 120.123909 | 15956 |
| Algoritmo C | ICMP (padrão) | 750 | 1949431 | 120.129781 | 16227 |
| Algoritmo C | ICMP (padrão) | 1000 | 1951990 | 120.001019 | 16266 |
| Algoritmo C | UDP | 500 | 1969605 | 121.282347 | 16374 |
| Algoritmo C | UDP | 750 | 1972542 | 120.270023 | 16400 |
| Algoritmo C | UDP | 1000 | 1958660 | 120.397113 | 16268 |

Tabela 3 – resultados do algoritmo C

Este algoritmo foi criado possibilitando diferentes formas de execução, permitindo listas e alterar portas de origem e destino, qual protocol utilizer (ICMP ou UDP), tamanho dos pacotes, inserir um arquivo contendo o pacote de dados, determinar o número de pacotes a ser enviado e adicionar um tempo entre cada pacote enviado. Entretanto, a quantidade de pacotes processados ainda não foi grande, principalmente com o protocol UDP, como pode ser visto na Fig. 5.




(a) Pacotes ICMP do algoritmo C




(b) Pacotes UDP do algoritmo C

Figura 5 – pacotes capturados pelo *Wireshark* no algoritmo C

A maioria dos pacotes, tanto ICMP quanto UDP, não foram processados pelo servidor, resultando emu ma baixa quantidade de pacotes processados.

- Visão geral dos resultados

Feitos os testes dos algoritmos, foram coletadas informações com relação à quantidade de pacotes enviados, tempo de execução e tipo de protocol utilizando, representados na Tabela 4 a seguir. Nesta tabela foi utilizando os resultados utilizando a função *fast_rand()* no algoritmo A.

| | Protocolo | Tamanho da carga (em bytes) | Pacotes enviados | Tempo de execução (em segundos) | Pacotes / segundo |
|---|---|---|---|---|---|
| Algoritmo A | ICMP | 500 | 3878411 | 180.539976 | 21482 |
| Algoritmo A | ICMP | 750 | 1841876 | 120.104044 | 15335 |
| Algoritmo A | ICMP | 1000 | 1971111 | 180.222469 | 10937 |
| Algoritmo B | TCP | 500 | 4679679 | 31.614147 | 148024 |
| Algoritmo B | TCP | 750 | 4656135 | 31.442411 | 148084 |
| Algoritmo B | TCP | 1000 | 4469497 | 38.215156 | 116956 |
| Algoritmo C | ICMP (padrão) | 500 | 1916743 | 120.123909 | 15956 |
| Algoritmo C | ICMP (padrão) | 750 | 1949431 | 120.129781 | 16227 |
| Algoritmo C | ICMP (padrão) | 1000 | 1951990 | 120.001019 | 16266 |
| Algoritmo C | UDP | 500 | 1969605 | 121.282347 | 16374 |
| Algoritmo C | UDP | 750 | 1972542 | 120.270023 | 16400 |
| Algoritmo C | UDP | 1000 | 1958660 | 120.397113 | 16268 |

Tabela 4 – resultados dos testes dos algoritmos

Analisando os resultados foram feitas comparações utilizando as variáveis **Pacotes Enviados** e **Pacotes / segundo**, levando em consideração as cargas de tamanho 1000 (bytes).

Fazendo a comparação com a variável **Pacotes Enviados**, constatou-se que o algoritmo B, apresenta um resultado 2.26 vezes mais eficiente que o algoritmo A e 2.28 vezes mais eficiente que o algoritmo C tanto com ICMP quanto UDP.

Da mesma forma, com a variável **Pacotes/segundo**, o algoritmo B apresenta um resultado 10.69 vezes mais eficiente que o algoritmo A e 7.19 vezes mais eficiente que o algoritmo C tanto com ICMP quanto UDP.

Os tamanhos das cargas de dados serviram para ter uma noção do desempenho de cada algoritmo. Esses tamanhos foram adicionados aos tamanhos totais dos pacotes somados com o tamanho das estruturas dos protocolos utilizados em cada algoritmo. Os tamanhos das cargas dos pacotes foram variados para obter uma noção do desempenho de cada algoritmo com cargas diferentes. Como pode-se analisar, o algoritmo B se destacou no quesito quantidade de pacotes por segundo. Por conta disso, o tempo de execução foi reduzido por limitação da máquina em armazenar todos os pacotes capturados.

## Conclusões

O ataque de negação de serviço consiste em consumir os recursos de uma determinada aplicação, deixando ela inoperante. Dentro dessa técnica foi analisado o funcionamento do processo de inundação do ataque. Atualmente esta técnica causa muitos danos a empresas, dificultando o acesso de usuários legítimos a elas.

Considerando o aumento de ocorrências de ataques do tipo inundação de dados em aplicações, o projeto permitiu compreender melhor os métodos DoS e DDoS quando aplicadas na área de segurança da informação. Acresenta-se ainda que foi possível colaborar com o conhecimento da técnica e posteriormente contribuir para a melhora de futuros ataques.

Neste trabalho foram abordadas as variações do ataque de negação de serviço, focando especificamente no funcionamento da técnica de inundação de sistemas utilizando pacotes ICMP *echo request*, TCP SYN e UDP. O objetivo do projeto foi elaborar um estudo sobre tráfego de redes gerado pelo ataque de inundação.

As tabelas com os resultados dos algoritmos implementados no projeto garantiram uma visão da eficiência que cada um dos ataques pode produzir, utilizando cada uma das variações apresentadas.

Considerando que os testes foram realizados em 40 máquinas, não é possível dizer se os algoritmos são capazes de inundar um alvo independente do protocolo que esteja sendo utilizado, apesar deles terem apresentado resultados satisfatórios com relação a quantidade de pacotes enviados por segundo. Portanto, o projeto proporcionou conhecimento sobre ataques de negação de serviço e possibilitou o desenvolvimento de algoritmos de inundação com pacotes ICMP, TCP e UDP.

Ataques de negação de serviço continuam sendo um problema e evoluem a cada dia. Diversos estudos são voltados para essa área complexa. Acredita-se que os resultados gerados nesse projeto possam auxiliar futuramente no desenvolvimento de trabalhos mais complexos e que explorem mais a fundo os recursos disponibilizados pelas tecnologias existentes.

## Referências bibliográficas


AUTRAN, Felipe. **DDoS: GitHub sofre maior ataque de negação de serviço da história**. Disponível em: https://www.tecmundo.com.br/seguranca/127777-ddos-github-maior-ataque-negacao-servico-historia.htm

BONGIOVANNI, Wilson. **Análise da Aplicação do Algoritmo de Viterbi na Detecção de Ataques Distribuídos de Negação de Serviço** – Instituto de Pesquisas Tecnológicas do Estado de São Paulo – 2014.

BRITO, Edivaldo. ***Wireshark*: capture dados e veja informações detalhadas da rede**. Disponível em: https://www.techtudo.com.br/tudo-sobre/*Wireshark*.html

COMANDO PING – IBM Knowledge Center – 2017, Disponível em: https://www.ibm.com/support/knowledgecenter/pt-br/POWER8/p8hcg/p8hcg_ping.htm

CICHONSKI, Paul; MILLAR, Tom; GRANCE, Tim; Karen, Scarfone. **Computer Security Incident Handling Guide** – National Institute of Standards and Technology – 2012.

FILHO, José. G. P. **O Protocolo IP** – Universidade Federal do Espírito Santo – 2015a.

FILHO, José. G. P. **O Protocolo ICMP** – Universidade Federal do Espírito Santo – 2017.

FILHO, José. G. P. **O Protocolo TCP** – Universidade Federal do Espírito Santo – 2015b.

GOMES, Lucas. C; ARAUJO, Marcos. S. A; CAMPOS, Vinícius. S. **Negação de Serviço e Botnets –** Universidade Federal do Rio de Janeiro – 2015.

KUROSE, James F; ROSS, Keith W. **Redes de computadores e a Internet: uma abordagem top-down** – 6ª ed. Pearson Education do Brasil, 2013.

LAUFER, Rafael. **Rastreamento de Pacotes IP Contra Ataques de Negação de Serviço –** Universidade Federal do Rio de Janeiro – 2005.

MAIWALD, Eric. **Network security: a beginner's guide** – 3rd ed. New York: McGraw-Hill, 2013.

MONTEIRO, Miguel. P. **O Sistema Operativo UNIX** – Faculdade de Engenharia da Universidade do Porto – 2011.

OWENS, Christopher. **Fast Random Number Generator on the Intel Pentium 4 Processor** – Intel Corporation, Folsom: California – 2012. Disponível em: https://software.intel.com/en-us/articles/fast-random-number-generator-on-the-intel-pentiumr-4-processor.

SAB, Gabriel; FERREIRA, Rafael; ROZENDO, Rafael. **Negação de Serviço, Negação de Serviço Distribuídas e Botnets**. Universidade Federal do Rio de Janeiro – 2013.

SANTOS, Uélinton. **Ataques Distribuídos de Negação de Serviço – Análise do Problema, Prevenção e Combate** – Instituto de Pesquisas Tecnológicas do Estado de São Paulo – 2004.

STALLINGS, William. **Criptografia e Segurança de redes: Princípios e práticas**. 4ed., São Paulo: Pearson Prentice Hall, 2008.

STALLINGS, William; BROWN, Lawrie. **Computer Security: principles and practice**. University of New South Wales, Australian Defence Force Academy – Third Edition.

SAUDE, Pedro. **O protocolo IP** – CCM – 2012. Disponível em: https://br.ccm.net/contents/276-oprotocolo-ip

UDP FLOOD ATTACK – Cloudflare - 2018?, Disponível em: https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/