UNIVERSIDADE
BEIRA INTERIOR

Detection of Stealthy Distributed Denial of Service Attacks Using Artificial Intelligence Methods

Vinícius de Miranda Rios

Tese para obtenção do Grau de Doutor em
Engenharia Informática
(3º ciclo de estudos)

Orientador: Prof. Doutor Mário Marques Freire
Co-orientador: Prof. Doutor Damien Magoni

Júri:
Prof. Doutor Luís F. B. A. Alexandre
Prof. Doutor Damien Magoni, Université de Bordeaux
Prof. Doutor Claudio de Castro Monteiro
Prof. Doutor Henrique João Lopes Domingos
Prof. Doutora Isabel C. C. S. Praça Gomes Pereira
Prof. Doutor Pedro Ricardo Morais Inácio
Prof. Doutor Tiago José dos Santos Martins da Cruz
Prof. Doutor Bruno Miguel Correia da Silva
Prof. Doutor João Carlos Raposo Neves

28 de fevereiro de 2025

Declaração de Integridade

Eu, Vinícius de Miranda Rios, que abaixo assino, estudante com número de inscrição D1361 do curso de Terceiro Ciclo de Estudos em Engenharia Informática da Faculdade de Engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridade da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, e que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assim assumo na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã, 14/03/2025

gov.br
Documento assinado digitalmente
VINICIUS DE MIRANDA RIOS
Data: 16/03/2025 10:06:38-0300
Verifique em https://validar.iti.gov.br

Vinícius de Miranda Rios

Thesis prepared at the Instituto de Telecomunicações – Delegação da Covilhã and at the Department of Computer Science of the University of Beira Interior and submitted to the University of Beira Interior for discussion in public session to obtain the Ph.D. Degree in Computer Science and Engineering.

This work has been funded by Portuguese FCT/MCTES through national funds and, when applicable, co-funded by EU funds under the project UIDB/50008/2020, and by operation Centro-01-0145-FEDER-000019 - C4 - Centro de Competências em Cloud Computing, co-funded by the European Regional Development Fund (ERDF/FEDER) through the Programa Operacional Regional do Centro (Centro 2020). This work has also been funded by CAPES (Brazilian Federal Agency for Support and Evaluation of Graduate Education) within the Ministry of Education of Brazil under the grant contract BEX 9095-13-6/2013 of the doctoral student Vinícius de Miranda Rios at the University of Beira Interior.

Cofinanciado por:

Dedication

I dedicate this thesis to the Lord God and the Lord Jesus Christ. I also dedicate it to my wife, Núbia Stefânia Alves Pereira; my parents, Rui Adelino Rodrigues Rios and Míriam Lúcia Miranda Rios; my brother, Rui Felipe de Miranda Rios; my dear grandmother, Maria Luiza Piedade Rodrigues Rios (in memoriam); my aunt, Ana Maria Rodrigues Rios de Almeida Fernandes; my cousins, Sofia Inês Rios de Almeida Fernandes, Rui Albertino de Almeida Fernandes, Normanda Maria Almeida Rodrigues and Analuria de Almeida Rodrigues; my stepmother and friend, Nathália Brito; and to all my family and friends who have supported me in one way or another to reach this milestone.

Acknowledgements

To professors Claudio de Castro Monteiro and Igor Yepes, my predecessors in the teaching profession and great friends, I would like to express my deepest gratitude. It was alongside you that everything began, and without the friendship and companionship we shared, I wouldn't be here today. Thank you for being more than just mentors, for being true friends and guides on this challenging and enriching journey. I will be forever grateful for everything you've done for me, both in my professional and personal life. May our friendship continue to flourish, and may we continue to support each other in all moments of life. Thank you, from the bottom of my heart, for being an essential part of my journey.

I want to express my gratitude for all the support I received during my Ph.D. studies from Instituto Federal de Educação, Ciência e Tecnologia do Tocantins (IFTO) and the professors of the Internet Systems course: Ana Paula Alves Guimarães de Col, Bruno Viana Coutinho, Carlos Henrique Corrêa Tolentino, Paulo Paz, Fagno Alves Fonseca, Francisco das Chagas de Sousa, Francisco Willians Makoto Placido Hirano, Helder Cleber Almeida Pereira, Mauro Henrique Lima de Boni, Napoleão Póvoa Ribeiro Filho, and Thiago Guimarães Tavares. They were essential figures in providing support and encouragement to ensure the completion of this doctorate.

To my journey companions in Covilhã, Portugal, Manoel Campos da Silva Filho, Breno Farias da Silva, Raysa Oliveira, Vanice Canuto Cunha, Musa Gwani Samaila and wife, Samila Valentin Bonilha, Wagner Quintanilha, Tiago Rosado, Tiago Simões, Paulo Cezar Cagliari (in memoriam) and to all the friends who made my days there more special, I would like to express my immense gratitude. Every shared moment, every exchanged smile, and every challenge overcome alongside you made my stay in Covilhã a truly unforgettable experience.

To my companion on this long doctoral journey, friend, and colleague, Manoel Campos da Silva Filho, for all the support and advice given to me so that I could move forward with determination and confidence. Your constant support made the challenges lighter and the achievements more significant.

To my lifelong friends who accompanied me on my doctoral journey, Alex Coelho, João Ricardo da Silva, Carlos Henrique Tolentino, Marcelo Perim, Márcio Aguiar, Eliomar Pinheiro Lima, Eliana Aires da Silva, Arthur Passos, Renato de Oliveira Bastos, Antônio Carlos Silveira, Helder Cleber Almeida Pereira, Mauro Henrique Lima de Boni, Marques Queiroz, and all the other friends, this dedication is a sincere expression of my gratitude and appreciation for each of you. May our friendship continue to strengthen over time.

I would also like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível (CAPES) for funding this thesis through the grant contract BEX 9095/13-6 and the University of Beira Interior, the Instituto de Telecomunicações of Portugal, and the Portuguese FCT/M-

CTES which are partners in this work.

To my dear doctoral supervisor, Mário Marques Freire, my dear co-supervisor, Damien Magoni, and the esteemed professor Pedro R. M. Inácio, I would like to dedicate these words of profound gratitude. Thank you for being the guides and mentors on this challenging and rewarding journey. Your guidance, support, and wisdom were instrumental to my academic and personal growth. Every piece of advice, every correction, and every encouragement were like beacons lighting the path to success. Your commitment and dedication to sharing knowledge and experience are truly inspiring. May we continue to collaborate and learn from each other. Thank you for believing in me and for being sources of inspiration throughout this journey. May our friendship and collaboration endure beyond the academic horizons.

I thank everyone that directly or indirectly helped me somehow to conclude this journey.

Resumo

Os ataques de negação de serviço têm sido usados para causar danos aos mais variados tipos de atividades fornecidas online. A grande quantidade de tráfego de ataque distribuído enviado ao alvo tem permitido criar assinaturas ou perfis de comportamento que possibilitaram o desenvolvimento de mecanismos de deteção para a mitigação destes. No entanto, à medida que novos formatos de ataques vão surgindo, por exemplo, ataques com baixa taxa de transmissão de tráfego, denominados de ataques *low-rate Denial of Service (DoS)*, novos mecanismos de deteção precisam de ser criados para combater eficazmente estas ameaças em evolução.

Muitos mecanismos de deteção baseiam-se essencialmente em análise estatística para identificar ataques *low-rate* DoS no tráfego de dados. No entanto, estes métodos apresentam frequentemente uma elevada taxa de falsos negativos e são aplicáveis apenas a dados em pequena escala. As técnicas de inteligência artificial têm sido amplamente utilizadas em várias áreas, incluindo análise de redes sociais e monitorização de doenças, e têm vindo gradualmente a ganhar destaque nos últimos anos no campo da cibersegurança.

Esta tese foca-se no estudo e no desenvolvimento de mecanismos de deteção que tenham um desempenho eficaz contra dois tipos específicos de ataques *low-rate* DoS: o ataque de Redução da Qualidade de Serviço (*Reduction of Quality (RoQ)*) e o ataque Slowloris. Para o ataque RoQ, estudamos o formato de transmissão do tráfego para criar um que seja similar, já que não se encontra disponível na internet software capaz de gerar tráfego deste tipo de ataque. Para o ataque Slowloris, foi utilizado software, disponível para *download*, desenvolvido especificamente para este fim. Posteriormente, analisámos o tráfego de ambos os ataques e extraímos características (*features*) que pudessem ser utilizadas pelos mecanismos de deteção.

Nesta tese foram desenvolvidas duas abordagens para classificação e deteção de ataques RoQ e Slowloris: a primeira abordagem é baseada na utilização separada de um conjunto de algoritmos de aprendizagem automática tradicionais, e a segunda abordagem é baseada em lógica difusa (fuzzy logic), um algoritmo de aprendizagem automática tradicional (que previamente tenha conduzido a bons resultados de classificação) e distância Euclidiana. Para a deteção do ataque RoQ, a primeira abordagem usa separadamente onze algoritmos de aprendizagem automática, nomeadamente *K-Nearest Neighbors (K-NN)*, *Multilayer Perceptron Neural Network (MLP)*, *Support Vector Machine (SVM)*, *Multinomial Naive Bayes (MNB)*, *Gaussian Naive Bayes (GNB)*, *Decision Tree (DT)*, *Random Forest (RF)*, *Gradient Boosting (XGB)*, *Logistic Regression (LR)*, AdaBoost e *Light Gradient Boosting Machine (LGBM)*, enquanto que a segunda abordagem consiste no método proposto por nós, o qual combina lógica difusa, o algortimo MLP e o método de distância Euclidiana. Para a deteção do ataque Slowloris, a primeira abordagem utiliza nove algoritmos de aprendizagem automática, nomeadamente K-NN, GNB, MLP, SVM, DT,

MNB, RF, XGB e LGBM, enquanto que a segunda abordagem consiste no nosso método, o qual combina lógica difusa, o algortimo RF e o método de distância Euclidiana. Ambas as abordagens utilizaram características (*features*) previamente selecionadas para classificar o tráfego de dados como tráfego de ataque ou tráfego legítimo. Os resultados obtidos mostraram que alguns algoritmos de aprendizagem automática (nomeadamente MLP e RF) assim como o nosso método, baseado em lógica difusa, um algoritmo de aprendizagem automática e distância Euclidiana são bons candidatos para serem utilizados para classificar ataques RoQ e Slowloris, mas a segunda abordagem com um tempo de execução ligeiramente maior para os detetar.

Palavras-chave

Algoritmos de Aprendizagem Automática, Ataque de Baixa Taxa (*Low-Rate*), Ataque de Negação de Serviço de Baixa Taxa, Ataque de Redução de Qualidade, Ataque Distribuído de Negação de Serviço de Baixa Taxa, Ataque Lento (*Slow*) de DoS, Ataque Slowloris, Distância Euclidiana, Lógica Difusa, Redes Neuronais.

Resumo Alargado

Ao longo dos anos, os serviços fornecidos por empresas ou instituições públicas, através da Internet, têm sido frequentemente alvo de inúmeros tipos de ataques cibernéticos. Dentre estes, os ataques de negação de serviços - cujo objetivo é bloquear qualquer tipo de tentativa de acesso aos serviços *online* - estão entre os mais perigosos, pelo facto de não possuírem ainda uma defesa efetiva contra a maioria deles, havendo apenas meios de minimizar a sua ação. Estes ataques, comumente chamados de ataques DoS, eram, inicialmente, desferidos apenas de uma única máquina atacante em direção à máquina alvo, utilizando pacotes mal formatados. Com o surgimento de defesas efetivas contra este tipo de ataque, os atacantes então redefiniram o formato, passando a utilizar um formato distribuído, ou seja, vários computadores atacantes transmitindo uma gigantesca quantidade de tráfego de dados em direção à máquina alvo. Desta forma, este tipo de ataque passou a ser denominado de ataque de Negação de Serviço Distribuído (*Distributed Denial of Service (DDoS)*). Este tipo de ataque tem a capacidade de causar danos mais abrangentes nos serviços online, tais como bloquear recursos dos *links* de conexão, de aplicações e/ou de *hardware*. Entretanto, com a evolução dos mecanismos de deteção mais robustos e efetivos também contra esta nova modalidade, uma maioria significativa de ataques passaram a ser ineficazes, necessitando que os atacantes reinventassem o modo como os serviços online pudessem sofrer algum tipo de dano. Nesse sentido, houve uma mudança de paradigma, pois a intenção agora é atacar o alvo de uma forma mais furtiva ou silenciosa. A ideia é enviar uma quantidade de tráfego de dados suficiente de modo a que passe desapercebido pelas ferramentas de defesa contra ataques do tipo DDoS, mas ao mesmo tempo, com o poder de danificar os serviços fornecidos pelas aplicações nos servidores. Assim, os ataques com esta característica passaram a ser denominados de ataques de Negação de Serviço de Baixa Taxa (*Low-rate Denial of Service (LDoS)*). Basicamente, estes podem reduzir a qualidade do serviço das aplicações ou, no limite, bloqueá-las.

Este trabalho foca-se na deteção dos ataques do tipo LDoS, mais especificamente nos ataques RoQ e Slowloris, com o objetivo de produzir uma abordagem que possa ser eficiente na deteção destes ataques. O trabalho de investigação aqui descrito iniciou-se com o estudo do ataque RoQ, verificando, primeiramente, como o tráfego deste tipo de ataque se comporta para causar danos à qualidade do serviço da aplicação alvo. O formato do tráfego possui picos moderados de fluxo de transmissão de pacotes de dados, seguido por um período de silêncio, alternando entre estes estados até a qualidade de serviço da aplicação se ir reduzindo até se aproximar eventualmente da negação de serviço. Nos picos de transmissão do tráfego de dados, o valor da taxa de transmissão não pode ultrapassar a largura de banda do alvo, cujo objetivo é evitar mecanismos de deteção de ataques DDoS. Diante disto, a fim de reproduzir este formato, procurou-se um *software* que pudesse gerar o fluxo de tráfego no padrão acima mencionado. Dentre os pacotes de *software* pesquisados, o que mais se destacou foi o Hping3, por possuir vários formatos

de tráfego, com taxas de transmissão de dados variando de baixa até alta. No entanto, este *software* não possui alguma opção para executar uma transmissão de fluxo de dados no formato do ataque RoQ. Por isso, foi desenvolvido um script em shell, designado por *Manipulated-RoQ (M-RoQ)* que manipula a saída do fluxo de transmissão do tráfego de dados do *software* Hping3 para o formato de transmissão do fluxo de dados no padrão do ataque RoQ.

Após a análise do tráfego gerado pelo M-RoQ num ambiente controlado, foram selecionadas três características (features) que possuíam valores capazes de permitir ao *software* de deteção a classificar o fluxo de tráfego como tráfego de ataque ou tráfego legítimo. A primeira característica escolhida foi o número de pacotes, visto que quando há um ataque, observou-se que a taxa de pacotes por segundo que chega ao alvo aumenta consideravelmente, quando comparado com a do tráfego legítimo. A segunda característica considerada foi o tempo médio entre chegadas de pacotes, já que, como o *software* de ataque inunda a rede com milhares de pacotes, estes não aguardam na fila de transmissão ao serem enviados pela placa de rede, pelo que são transmitidos o mais rápido possível, diminuindo o valor do atraso médio entre chegadas de pacotes. A terceira característica considerada foi a entropia do quinteto composto por endereço IP de origem, porta de origem, endereço IP de destino, porta de destino e protocolo de transporte (TCP ou UDP) para o tráfego de rede. A relevância desta característica resulta do facto de o tráfego gerado pelo *software* Hping3 desestabilizar o alvo, utilizando um vasto leque de protocolos e endereços *Internet Protocol (IP)*s aleatórios. Com isto, o quinteto atrás referido deverá ter um valor de entropia elevado, já que quanto mais variados são os valores de uma dada variável, mais elevado será o valor da respetiva entropia. Para a abordagem da deteção do ataque RoQ, foram selecionados métodos que fossem capazes de fazer uma classificação baseada em padrões que pudessem diferenciar o tráfego de ataque do tráfego de utilizadores legítimos. Desta forma, a primeira abordagem para a deteção deste tipo de ataques consistiu na utilização separada de um conjunto de algoritmos de aprendizagem automática, nomeadamente, K-NN, MLP, SVM, MNB, GNB, DT, RF, XGB, LR, AdaBoost, e LGBM. A segunda abordagem consistiu num método que usa lógica difusa, devido à sua elevada capacidade de classificação baseada em regras pré-definidas. Este segundo método envolve também a utilização de um algoritmo de aprendizagem automática, sendo escolhido aquele que apresentar melhores resultados de classificação dentre os algoritmos de aprendizagem automática utilizados nos testes de deteção deste tipo de ataques. Por fim, foi utilizada a distância Euclidiana como medidor da distância entre os valores dos atributos já classificados entre os dois métodos anteriores, a fim de escolher o melhor entre eles.

A seguir, procurou-se na internet *traces* de tráfego que contivessem tráfego de ataque RoQ para ser utilizado como base de treino para os métodos de aprendizagem automática. Como nenhum *trace* de tráfego foi encontrado especificamente para este ataque, foi utilizado um genérico, ou seja, um que possuísse tráfego de ataque distribuído com taxa de transmissão moderada.

Além disto, optou-se, para a criação dos *traces* de tráfego para treino e teste, por ambientes que pudessem reproduzir o mais próximo do real, bem como ambientes reais, todo o processo de ataque e deteção, não usando nenhum *software* de simulação. Para o ambiente emulado, utilizou-se o *software* netkit. Este, utiliza um ambiente virtualizado, emulando computadores e simulando a infraestrutura da rede. Para o ambiente real, foi utilizada a infraestrutura da rede do IFTO visando ter um ambiente o mais próximo possível do real. Com os *traces* de tráfego gerados e coletados (tráfego de ataque e legítimo) nos ambientes acima citados, foi necessário transformá-los em valores que serão representados por cada característica selecionada anteriormente. Para o computo destes valores, o *trace* de tráfego de cada ambiente foi fatiado em amostras de um segundo cada (com exceção da última que pode ter uma duração inferior a um segundo), sendo a extração dos dados feita no fluxo de dados do respetivo instante de tempo. No final deste processo, foi gerado um dataset para cada *trace* de tráfego, emulado e real. O valor computado para a característica número de pacotes foi feito somando-se todas as transmissões feitas no espaço amostral. O valor computado para a característica tempo médio entre chegadas dos pacotes foi computado fazendo uma subtração entre o respetivo valor do pacote subsequente e do anterior e assim sucessivamente até ao final da amostra. Posteriormente, estes valores da operação de subtração são somados e é feita uma média aritmética entre eles. O valor computado para a característica entropia do quinteto foi obtido através da extração da respetiva informação a partir do tráfego da amostra: IP de origem, porta de origem, IP de destino, porta de destino e protocolo de transporte. Desta forma, foi calculada a variabilidade dos valores destes campos do endereço IP de cada transmissão distinta dentro do fluxo amostral. Portanto, cada dataset será composto de várias tuplas, em que, cada tupla é composta de 3 valores provenientes das 3 características acima mencionadas.

Antes da operação de deteção ser executada, foi preciso fazer a rotulagem dos dados das tuplas do dataset genérico para que ao ser utilizado pelos algoritmos de aprendizagem automática para rotular os dados das tuplas dos datasets dos ambientes emulado e real, estes possam saber a priori quais tipos de dados possuem valores que se assemelham com um tráfego de ataque e quais se assemelham com um tráfego legítimo. Após isto, a abordagem da deteção se inicia visando classificar os datasets gerados pelos ambientes emulado e real. A primeira classificação do tráfego é feita pelos algoritmos de aprendizagem automática da primeira abordagem e, posteriormente, pela lógica difusa seguida da classificação da rede neuronal *multilayer perceptron* da segunda abordagem. Cada tupla dos datasets é rotulada como ataque ou legítimo. No fim da classificação, é feita uma comparação entre os rótulos gerados pelos dois métodos. A diferença entre as classificações, ou seja, os valores da entropia e do número de pacotes da diferença entre os rótulos das tuplas de cada método, são armazenados. Em seguida, o método distância euclidiana é acionado e utiliza os valores armazenados (entropia e número de pacotes) de cada rótulo e compara-os com cada uma dos valores das tuplas (entropia e número de pacotes) de todo o dataset genérico do ataque RoQ. Assim, o atributo que possuir a menor distância

(aquele que possuir o menor valor de resultado) dentre os valores comparados no dataset genérico do ataque RoQ, substituirá com seu rótulo, caso o rótulo seja diferente, o rótulo do valor do dataset do ambientes emulado e real. Isto ocorrerá com todos os valores do resultado da diferença da comparação. Em seguida, é feita uma comparação dos rótulos da classificação da lógica difusa e da rede neuronal, juntamente com os novos rótulos gerados pelo método da distância euclidiana. Desta forma, havendo uma quantidade de rótulos de ataque superior a quantidade de rótulos legítimo, o resultado final do rótulo para aquela tupla do dataset será ataque, caso contrário, será legítimo e havendo um empate a tupla é marcada com um rótulo de *warning*. Este último rótulo é um indicativo de que algo não saiu conforme planejado e precisa ser verificado pelo administrador de rede para averiguar o que causou tal rotulação e, se caso houver anomalias, o tráfego desta amostra será bloqueado, caso contrário, será permitido seguir para o destino.

O melhor resultado de classificação de ataques RoQ usando algoritmos de aprendizagem automática foi obtido com o MLP, tendo atingido uma precisão (*accuracy*) de 98,97% em ambiente emulado e uma precisão de 99,92% em ambiente real, enquanto que a abordagem usando lógica difusa, MLP e distância Euclidiana conduziu a uma precisão de 99,36% em ambiente emulado e uma precisão de 100,00% em ambiente real.

Para a deteção do ataque Slowloris, todos os passos citados acima foram feitos, mas com algumas modificações:

- Primeiro: foi adicionado mais algoritmos de machine learning para a comparação com a abordagem de deteção criada;
- Segundo: os atributos escolhidos foram somente a entropia e a quantidade de portas distintas, a qual é calculada através do conjunto de transmissão de portas de origem contendo o tamanho do quadro e tipo de protocolo distintos;
- Terceiro: os ambientes de testes de desempenho utilizados foram três: o emulado, utilizando o netkit; a *Local Area Network (LAN)*, utilizando o ambiente local do IFTO, mantendo-se o mesmo esquema do ataque RoQ; e a rede *Metropolitan Area Network (MAN)*, reproduzindo o ataque utilizando a infraestrutura da Internet da cidade de Palmas/Tocantins/Brasil;
- Quarto: para a deteção do ataque, o processo foi o mesmo, com a diferença de que utilizou-se somente o atributo quantidade de portas distintas na parte de classificação dos dados com o rótulo de *warning*;
- Quinto: devido ao *overffiting* gerado por utilizar separadamente as base de dados geradas por cada ambiente, optou-se por fazer a união de todas elas em uma base de dados única. Desta forma, foi obtido um melhor desempenho pelos algoritmos de deteção.

O melhor resultado de classificação de ataques Slowloris usando algoritmos de apren-

dizagem automática foi obtido com o RF, tendo atingido uma precisão de 99,54% (com otimização de hiperparâmetros), enquanto que a abordagem usando lógica difusa, RF e distância Euclidiana conduziu a uma precisão de 99,80%.

Abstract

Distributed Denial of Service (DDoS) attacks have been used to disrupt various online activities. The significant traffic volume of these distributed attacks has enabled the identification of signatures and behavior profiles that fostered the development of detection mechanisms for mitigating these attacks. However, as new attack types emerge, such as low-rate Denial of Service (DoS) attacks, new detection mechanisms need to be developed to combat these evolving threats effectively.

Many detection mechanisms rely primarily on statistical analysis to identify low-rate DoS attacks in data traffic. However, these methods often exhibit a high rate of false negatives and are only applicable to small-scale data. Artificial intelligence techniques have been widely employed in various fields, including social network analysis and disease monitoring, and have gradually gained prominence in the field of cybersecurity in recent years.

This thesis focuses on studying and developing detection mechanisms that exhibit effective performance against two specific types of low-rate DoS attacks: the Reduction of Quality (RoQ) attack and the Slowloris attack. For the RoQ attack, we examine the traffic transmission format to create a similar one, as there is no existing software capable of generating this type of attack traffic on the internet. For the Slowloris attack, we utilized free and open-source software specifically developed for this purpose. Subsequently, we analyze the traffic from both attacks and extract features that can be used by detection mechanisms.

In this thesis, two approaches have been developed for classifying and detecting RoQ and Slowloris attacks: one approach is based on the separate use of a set of traditional Machine Learning (ML) algorithms and the second approach is based on fuzzy logic plus one traditional ML algorithm (that previously led to good classification results) and Euclidean distance. For the RoQ attack detection, the first approach uses eleven separate machine learning algorithms, namely K-Nearest Neighbors (K-NN), Multilayer Perceptron Neural Network (MLP), Support Vector Machine (SVM), Multinomial Naive Bayes (MNB), Gaussian Naive Bayes (GNB), Decision Tree (DT), Random Forest (RF), Gradient Boosting (XGB), Logistic Regression (LR), AdaBoost, and Light Gradient Boosting Machine (LGBM), while the second approach consists in our proposed method which combines fuzzy logic, the MLP algorithm, and the Euclidean distance method. For the Slowloris attack detection, the first approach utilizes nine machine learning algorithms, namely K-NN, GNB, MLP, SVM, DT, MNB, RF, XGB, and LGBM, while the second approach consists in our proposed method which combines fuzzy logic, the RF algorithm, and the Euclidean distance method. Both approaches utilize previously selected features to classify the data traffic as either *attack* traffic or *legitimate* traffic. The obtained results show that some ML algorithms (namely MLP and RF) as well as our approach based on fuzzy logic, one ML algorithm, and Euclidean distance are good candidates to be used to classify RoQ

and Slowloris attacks, but the latter approach with a slightly longer runtime for detecting them.

Keywords

Euclidean Distance, Fuzzy Logic, Low-rate Attack, Low-rate Denial of Service (LDoS) Attack, Low-rate Distributed Denial of Service (LDDoS) Attack, Machine Learning Algorithms, Neural Networks, Reduction of Quality RoQ Attack, Slow DoS Attack, Slowloris Attack.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

ACL	Access Control List
ADDoS	Advanced Distributed Denial of Service
AFM	Analysis and Filtering Module
ALP	Application Layer Protocols
AOMDV	Ad hoc On-demand Multipath Distance Vector
APSO	Adaptive Particle Swarm Optimization
AQM	Active Queue Management
AUC	Area Under the ROC Curve
BA-BNN	Bat Algorithm associated with a BP Neural Network
BGP	Border Gateway Protocol
Botnet	roBot Networks
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível
CBR	Constant Bit Rate
CCID	Cross-Correlation Identity Distinction
CDA	Changepoint Detection Algorithm
CE	congestion experienced
CM	Collection Module
CoA	Center-of-Area
CORE	Common Open Research Emulator
CPR	Congestion Participation Rate
CPU	Central Processing Unit
CTS	Clear to Send
CUSUM	Cumulative Sum
cwnd	congestion window
DDoS	Distributed Denial of Service

DFT	Discrete Fourier Transform
DNS	Domain Name System
DoS	Denial of Service
DPM	Deterministic Packet Marking
DSR	Design Science Research
DT	Decision Tree
DTW	Dynamic Time Wrapping
ED	Euclidean Distance
EEMD	Ensemble Empirical Mode Decomposition
EWMA	Exponential Weighted Move Average
FEC	Forward Error Correction
FL	Fuzzy Logic
FMIPv6	Fast Mobile Internet Protocol version 6
FMON	Flow Monitoring
FMT	Flow Monitoring Table
FN	False Negative
FP	False Positive
FPR	False Positive Rate
FRRED	Fair Robust Random Early Detection
FTP	File Transfer Protocol
GA	Genetic Algorithm
GE	Generalized Entropy
GID	Generalized Information Divergence
GNB	Gaussian Naive Bayes
HAWK	Halting Anomaly with Weighted Choking
HGB-FP	Histogram-based Gradient Boosting and Finding Peaks
HHT	Hilbert-Huang Transform

HTTP	HyperText Transfer Protocol
IBGE	Instituto Brasileiro de Geografia e Estatística
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IFTO	Instituto Federal de Educação, Ciência e Tecnologia do Tocantins
IMF	Intrinsic Mode Function
IoT-SDN	Internet of Things-Software Defined Networks
IP	Internet Protocol
IRC	Internet Relay Chat
IRTQB	Improved Random Time Queue Blocking
ISP	Internet Service Provider
K-NN	K-Nearest Neighbors
KPCA	Kernel Principal Component Analysis
LAN	Local Area Network
LDoS	Low-rate Denial of Service
LGBM	Light Gradient Boosting Machine
LoRDAS	Low-Rate DoS attacks against Application Servers
LR	Logistic Regression
LS-ITSVM	Locality Sensitive-Incremental Transductive Support Vector Machine
MAC	Media Access Control
MAD	Modeled Attack Detector
MAN	Metropolitan Area Network
MANET	Mobile Ad hoc Network
MIPv4	Mobile Internet Protocol version 4
MIPv6	Mobile Internet Protocol version 6
ML	Machine Learning
MLP	Multilayer Perceptron Neural Network

MLP-GA	Multilayer Perceptron with a Genetic Algorithm
MNB	Multinomial Naive Bayes
M-RoQ	Manipulated-RoQ
NCAS	Normalized Cumulative Amplitude Spectrums
NoTuLA	Novel Tuneable Low-Intensity Adversarial Attack
PAD	Periodic Attack Detector
PCA	Principal Component Analysis
PDoS	Pulsing Denial-of-Service
PDU	Protocol Data Unit
PF	Performance and Features
QCD	Quickest Changepoint Detection
QDA	Quadratic Discriminant Analysis
QoS	Quality of Services
RAI	Random Answer Instant
RAM	Random Access Memory
RBF	Radial Basis Function
RED	Random Early Detection
RED-PD	Random Early Detection-Preferential Dropping
RF	Random Forest
RoQ	Reduction of Quality
RRED	Robust RED
RREQ	Route REQuest
RST	Random Service Time
RTO	Retransmission TimeOut
RTQB	Random Time Queue Blocking
RTS	Request to Send
RTT	Round Trip Time

RUDY	R-U-Dead-Yet
SADBSCAN	Self-Adaptive Density-Based Spatial Clustering of Applications with Noise
SBID	Statistical Based Intrusion Detection
SDN	Software-Defined Networking
SDToW	Slowloris Detecting Tool for WMNs
SEDP	Spectral Energy Distribution Probability
SeVen	Selective Defense for Application Layer DDoS Attacks
SPE	Squared Prediction Error
SSM	Self-Similarity Matrix
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TDDoS	Traditional Distributed Denial of Service
TN	True Negative
TP	True Positive
TS-SVM	Two-step Self-adjusting Support Vector Machine
TSVM	Transductive Support Vector Machine
UDP	User Datagram Protocol
VANETs	Vehicular Ad hoc NETworks
VOIP	Voice Over Internet Protocol
WMN	Wireless Mesh Networks
WRED	Weighted RED
XGB	Gradient Boosting

Chapter 1

Introduction

1.1 Thesis Scope and Focus

Web-based services are in operation for over 30 years since the emergence of the world-wide web and they have been growing year after year, along with the evolution of personal equipment such as tablets, notebooks and, especially, smartphones that allow immediate access to web from nearly anywhere. In the world market, online sales surpassed 4.2 trillion dollars in the year 2020 [9]. According to Instituto Brasileiro de Geografia e Estatística (IBGE), the mobile device is the main mean of accessing the internet in Brazil, with a direct impact on online purchases, having a growth of 35% in 2017 [10] and reaching 55.1% in 2020 [11]. Also according to the study presented by Ericsson [12], the worldwide total mobile data traffic is estimated to grow by a factor of around 3 between 2023 and 2029. In addition, other online services such as games, social networks, video communication services, cloud storage services, chat applications, etc. have also had a relevant usage increase in recent years, especially during the coronavirus pandemic.

All this technology brings with it a myriad of cyberattacks, aiming different targets, using different techniques with different objectives. Among them, the main one is to disrupt internet services. A research made by Symantec revealed that Brazil is among the 3 countries that receives most of the cyberattacks in the world, with 9.8% [13] of all threats detected being made by internal agents. Many of these attacks were motivated by the switch to remote work made by many companies during the coronavirus pandemic, which generated an increase of 220% in the first half of the year 2021 compared to the same period of 2020 [14]. From the incidents reported, 11% were caused by denial of service attacks [15].

Denial of service attacks are a category of cyberattacks used to degrade or halt the service availability of the targeted providers, whether it is a company, a public agency, etc. The main focus of the disruption is to cause financial losses and to negatively impact the brand of the service provider. The first known DoS attack occurred in 1996 against an Internet Service Provider (ISP) called Panix [16]. The attack aimed at flooding the ISP server with several bogus packets simulating the Transmission Control Protocol (TCP) three-way handshake (3-way handshake), freezing it for approximately 36 hours. As online services have increased over time, these attacks have also increased and changed, becoming more aggressive and targeting new vulnerabilities. Compared with the first quarter of 2020, there was a 25% increase in daily attacks in 2021 [17], most likely due to the companies having to provide for remote workers. Since then, attacks have evolved,

becoming more sophisticated and causing more and more damage to service providers. DDoS attacks have continually grown in size and sophistication, but 2023 accelerated this trend at an unforeseen pace [18]. It wasn't just the number of DDoS attacks that increased in 2023, but also the variety of attack types that stood out from previous years.

The evolution of research in the area of DoS attack detection, has increased the capacity to identify, mitigate, or even stop many of them. Consequently, attackers have had to improve their attack procedures, making them more powerful yet stealthier. The Low-rate Denial of Service (LDoS) attacks have emerged as a new threat. These attacks are characterized by low levels of traffic with burst periods, aimed at overwhelming border routers and/or servers to make them drop legitimate packets [19]. The generated traffic bursts must be equal to or slightly lower than the victim's link bandwidth, allowing the attack to bypass known defense mechanisms [20]. This traffic shaping is necessary because the primary goal of LDoS attacks is not to stop the victim's service but to degrade the Quality of Service (QoS) of the applications, causing legitimate clients to lose interest in the service.

Among the low-rate attack types, there are those that specifically target web server applications, such as Apache. These attacks are classified as slow DoS attacks, and they usually spoof one or more fields of the HTTP header in the requests. The objective is to initiate multiple connections with bogus HTTP header traffic requests to the web server. The server interprets these connections as being slow, causing it to wait indefinitely for the client's response and thereby consuming server resources until it completely stops [21].

Technological advances create new vulnerabilities, which in turn create new types of LDoS attacks, as shown in [20, 21, 22, 23, 24]. In recent years, much work has been done to mitigate these attacks, although it is still in its infancy [25, 26, 27, 28, 29]. Therefore, more research is needed for contributing to the evolution of the required solutions against these new attacks.

The preference for the RoQ attack arises from its relatively simple implementation for disrupting applications utilizing TCP and User Datagram Protocol (UDP) protocols. This simple attack is attributed to the format of its traffic, which, despite its simplicity, has the potential to cause significant damage. Detecting these attacks presents a particular challenge due to their stealthy and intermittent nature, allowing them to blend seamlessly with normal traffic patterns. Similarly, the Slowloris attack has gained prominence as one of the most frequently used methods to incapacitate web server applications. Its widespread use is partly due to the availability of numerous tools, developed in various programming languages, that facilitate the launch of this attack. The insidiousness of the Slowloris attack lies in its ability to maintain open connections by sending partial requests or extremely slow data streams, making it exceptionally difficult to detect. This covert nature poses a significant challenge in distinguishing malicious traffic from legitimate requests, further complicating the task of ensuring network security. Therefore, this thesis addresses the problem of detecting RoQ attacks and Slowloris attacks.

1.2 Problem Statement

Protocols and operating systems are susceptible to programming failures, which allow vulnerabilities to be exploited by malicious users. As a result, serious damage can be inflicted on intermediate or end devices across the internet by targeting vulnerabilities in transport layer and application layer protocols.

Such vulnerabilities can include the following [30, 31, 32]:

- **TCP features:** the Retransmission Timeout function is the target of various LDoS attack types. The attacker sends periodic traffic bursts every 1 second to fill the queues of the routers, forcing the TCP protocol to adjust the transmission rate to the ”conditions” of the network. The result of this action is the reduction of the throughput, causing the gradual decrease of the application’s QoS;

- **Active Queue Management:** The droptail is the internet routers’ default Active Queue Management (AQM) algorithm, which has the worst packet queue management among AQM algorithms [33], making it the perfect target for some LDoS attack types. The attacker sends periodic traffic bursts to fill the queue of the routers forcing the TCP protocol to enter continuously in Retransmission TimeOut (RTO) mode, as well as forcing the UDP protocol to lose data packets, causing the gradual decrease of the application’s QoS;

- **HTTP header:** Some fields of the HTTP header can be changed with spoofed data in order to manipulate the correct execution of the HTTP connection request/reply protocol. Attackers can send bogus HTTP header requests or replies towards the webserver forcing it to wait endlessly for the request/reply, consuming its resources, and finally crashing the server.

A common element in exploiting these three vulnerabilities, is the use of low-rate flooding traffic. All LDoS attacks, including the RoQ attack and the Slowloris attack, employ low-rate flooding traffic directed at the victim. This is largely effective because these connection requests often bypass the detection barriers to reach their target. The challenge in detecting the RoQ attack lies in identifying subtle, yet intentional, degradations in service quality, typically achieved through intermittent bursts of traffic. These bursts are strategically designed to overload router packet queues, with the aim of hitting the retransmission timeout of the TCP protocol, thereby preventing it from recovering from the packet loss. This continuous triggering of retransmission timeouts leads to a situation where the TCP packet retransmission rate eventually drops to nearly zero. Conversely, detecting the Slowloris attack involves recognizing connections that are deliberately kept open through the transmission of partial requests or extremely slow data streams. This type of attack aims at depleting server resources by sustaining these connections, sending partial HTTP

requests with an encoding `\r\n` tag at the end of the header, ultimately leading to a denial of service.

The absence of distributed RoQ attack and Slowloris attack traffic traces for attack detection, as well as RoQ attack tools for traffic generation, poses a significant challenge. Most of the RoQ attack-related studies are conducted in simulated environments using testbeds, with the attack traffic generated solely within simulation software. Conversely, the Slowloris attack-related studies utilize self-gathered and/or public datasets, but these do not contain distributed traffic, as they rely on a single point of attack. Furthermore, the maximum accuracy achieved among RoQ and Slowloris attack detection related works is 99,0% [34] and 99.61% [35] respectively, suggesting potential for improvement. We propose to create distributed attack traffic datasets from different types of testbed scenarios and to investigate artificial intelligence methods to address false positive and false negative detection rate issues, which will be discussed in the following section.

1.3 Research Objectives

Following the problem definition described in the previous section, the objective of the research work presented in this thesis is to propose, implement, and critically evaluate: (i) various testbed environments, (ii) a novel approach alongside a suite of classifiers each leveraging machine learning algorithms.

To tackle the attack detection problem and enhance the accuracy rates of related works, two strategies were developed for both the RoQ and Slowloris attacks. For the RoQ attack, the first strategy involves utilizing a set of machine learning algorithms, while the second entails an approach employing a combination of fuzzy logic, MLP neural network, and the Euclidean distance method. Similarly, for the Slowloris attack, the first strategy employs a set of machine learning algorithms, and the second utilizes a combination of fuzzy logic, random forest, and the Euclidean distance method. Our approaches, proposed in this thesis, explore statistical features such as packet length, average inter-arrival time of packets, entropy, and packet ports of internet traffic flows. The performance of these classifiers is evaluated in terms of accuracy, recall, precision, and F1-score. Additionally, the efficiency of each approach is assessed in computational terms, including evaluations of CPU usage, execution time, and memory requirements. To support the objectives of this research, the following specific goals have been established:

- Presenting a literature and state-of-the-art review about LDoS attacks such as tools, attack types, and detection and defense mechanisms;
- Designing and implementing 3 types of lab testbeds (emulated, LAN and MAN) for classification of network traffic;

- Developing two datasets comprising distributed traffic from RoQ and Slowloris attacks and including legitimate traffic;
- Designing and implementing a script tool to launch the RoQ attack traffic in the format described by its creators;
- Selecting features with optimal performance in classifying network traffic as ”attack” or ”legitimate”;
- Designing and implementing a novel approach combining artificial intelligence methods;
- Analyzing the performance of the set of classifiers in terms of Precision, Recall, F-measure, and Accuracy and in terms of computational efficiency (execution time, and CPU and memory usage);
- Comparing the performance of our classifier approach to other classifiers, such as K-Nearest Neighbors, Gaussian Naive Bayes, Multi-layer Perceptron Neural Network, Support Vector Machine, Decision Tree, Multinomial Naive Bayes, Random Forest, Gradient Boosting and Light Gradient Boosting Machine.

1.4 Thesis Statement

This thesis proposes an innovative approach to detecting LDoS attacks in computer networks, utilizing a set of artificial intelligence classifiers such as machine learning algorithms including K-Nearest Neighbors, Gaussian Naive Bayes, Multi-layer Perceptron Neural Network, Support Vector Machine, Decision Tree, Multinomial Naive Bayes, Random Forest, Gradient Boosting, and Light Gradient Boosting Machine, as well as fuzzy logic, and the Euclidean distance metric. The thesis statement is as follows:

The data traffic generated by an LDoS attack presents unique characteristics that may enable its identification. In the RoQ attack, characteristics such as traffic bursts generated in a brief period of time, short average inter-arrival time, and high entropy values due to significant heterogeneity among packets in the same time frame were identified as features to be used in the classification approach. The classification approach involved a set of machine learning algorithms as well as our method based on the combination of fuzzy logic, MLP neural network, and Euclidean distance. Similarly, in the Slowloris attack, characteristics like burst data packets generated in a brief period of time and the presence of a large number of different destination ports were identified as features. These features were utilized in the classification approach, which employed a set of machine learning algorithms as well as our method based on the combination of fuzzy logic, random forest, and Euclidean distance. These approaches led to high accuracies in the classification of RoQ and Slowloris attacks.

1.5 Adopted Approach for Solving the Problem

The research methodology employed in this study is known as Design Science Research (DSR) [36]. This methodology encompasses a framework that offers guidance and direction regarding the anticipated outcomes of a research investigation. In applying this methodology, a systematic correlation of research activities was established to address the aforementioned problems. The procedures adopted include the following:

- **Survey:** The state-of-the-art about LDoS attacks covering all aspects related to the attack types, detection/defense mechanisms and location, testbed environments, and traffic approach should be surveyed, to allow us to select and investigate open issues;

- **Problem identification:** The problem identification process involves identifying the vulnerabilities exploited by RoQ and Slowloris attacks, creating distinct environments (emulated, LAN, and MAN) for generating distributed attack traffic, selecting the most effective features to achieve high accuracy, identifying (or creating) the proper attack tools to use for dataset production, and choosing and exploring suitable algorithms for the classification of the attacks, to improve the detection accuracy of these attacks reported in the publications of this specific area;

- **Testbed environments:** The environments selected to reproduce the entire procedure of the LDoS attacks should be created using an emulated software, a small-scale network (i.e., LAN network) and a MAN network;

- **Attack tools:** Due to the absence of any available RoQ attack tool, it is necessary to implement one. We will investigate the use of Hping3 within a shell script, which adjusts the timing between traffic bursts. For the Slowloris attack, an existing tool named `slowhttptest` will be explored;

- **Attack-mimicking experiments and production of datasets:** With the environments and attack software duly defined and created, experiments should be conducted to generate datasets containing both legitimate and attack traffic;

- **Classification approaches and their evaluation:** Two approaches will be investigated for classifying RoQ and Slowloris attacks: one approach based on the use of traditional ML algorithms and the other based on fuzzy logic plus one traditional

ML algorithm and Euclidean distance. The classification performance and resource usage of these strategies should be evaluated over the datasets with the RoQ and the Slowloris attacks.

- **Dissemination of results:** We expect to publish three papers, one devoted to a survey about the detection of slow attacks, a second paper exploring the application of the two strategies mentioned above for classifying RoQ attacks, and a third paper addressing those strategies for the classification of Slowloris attacks;

- **Writing of the doctoral thesis:** We explore the challenges and solutions related to LDoS attacks and the detection of these attacks through artificial intelligence methods. This thesis is organized into a series of chapters, based on the published papers, that address various aspects of the issue, from understanding low-rate DoS attacks to the application of artificial intelligence algorithms for their detection.

1.6 List of Scientific Publications

This Ph.D. thesis consists of the following research papers, published in peer-reviewed journals and conferences, resulting from the research work leading to this thesis:

1. **Detection and Mitigation of Low-rate Denial-of-Service Attacks: A Survey**, Vinícius de Miranda Rios, Pedro R. M. Inácio, Damien Magoni, Mário M. Freire. IEEE Access, vol. 10, pp. 76648–76668, 2022. DOI: 10.1109/ACCESS.2022.3191430 [29].

2. **Detection of Reduction-of-Quality DDoS Attacks Using Fuzzy Logic and Machine Learning Algorithms**, Vinícius de Miranda Rios, Pedro R. M. Inácio, Damien Magoni, Mário M. Freire. Computer Networks, Volume 186, pp. 1–18, 2021, Article 107792, Elsevier. DOI: 10.1016/j.comnet.2020.107792 [37].

3. **Detection of Slowloris Attacks Using Machine Learning Algorithms**, Vinícius de Miranda Rios, Pedro R. M. Inácio, Damien Magoni, Mário M. Freire. In Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing (SAC'24), Avila, Spain, pp. 1321-1330, 2024. DOI: 10.1145/3605098.3635919 [32].

Paper 1 [29] was published in a journal listed in Scimago Journal & Country Rank as Q1 at the time of its publication and, at the time of writing of this Ph.D. thesis, it has received

43 citations in Google Scholar. This paper appears with minor modifications in Chapter 2 of this thesis.

Paper 2 [37] was published in a journal listed in Scimago Journal & Country Rank as Q1 at the time of its publication and, at the time of writing of this Ph.D. thesis, it has received 91 citations in Google Scholar. This paper appears with several modifications in Chapter 3 of this thesis.

Paper 3 [32] was published in the proceedings of an international conference ranked as B in the CORE 2020 Conference Portal and, at the time of writing of this Ph.D. thesis, it has received 5 citations in Google Scholar. This paper appears with minor modifications in Chapter 4 of this thesis.

1.7 Main Scientific Contributions

This section briefly describes the main scientific contributions resulting from the research work presented in this thesis.

The first contribution of this thesis is an in-depth description of the approaches for detection/defense mechanisms and a comprehensive analysis review of the literature on LDoS attacks. This study is described in Chapter 2, which consists of a paper published in *IEEE Access* [29].

The second contribution consists of a detailed study of the RoQ attack and a proposed solution for its detection. A tool named M-RoQ was created to manipulate the attack traffic generated by Hping3 due to the lack of tools that could launch a RoQ attack traffic. The three chosen features, selected through the attack traffic analysis, were crucial for the detection mechanism to have a good detection performance. The classification approach included two different strategies, one based on the separate use of eleven traditional machine learning algorithms and the other based on the use of fuzzy logic, one traditional machine learning algorithm (that previously led to the best classification results), and Euclidean distance. These strategies allowed us to create a detection tool with high accuracy and low rates of false positives and false negatives. This study is described in Chapter 3, which is based on the paper published in *Computer Networks* [37].

The third contribution consists of a detailed study of the Slowloris attack and a proposed solution for its detection. The two chosen features, selected through the attack traffic analysis, were crucial in ensuring the high performance of the detection mechanism. The classification approach included two different strategies, one based on the separate use of nine traditional machine learning algorithms and the other based on the use of fuzzy logic, one traditional machine learning algorithm (that previously led to the best classification results), and Euclidean distance. These strategies allowed us to build a Slowloris detection tool with high accuracy and low rates of false positives and false negatives. This

study is described in Chapter 4, which consists of a paper published in the *ACM/SIGAPP Symposium on Applied Computing* conference proceedings [32].

1.8 Thesis Organization

This thesis is structured in five chapters. With the exception of the first and fifth chapters, which are devoted to the introduction and conclusions, each of the three main chapters is formed by research articles published in international venues. The subjects and organization of the main chapters of this thesis can be summarized as follows.

Chapter 1 describes the context of this thesis, explaining the scope and focus of the research work and presenting the problem addressed by the thesis and the objectives to be accomplished, as well as the thesis statement and the adopted approach for solving the problem. A summary of the main contributions of this thesis is also included, followed by the description of the organization and structure of the thesis.

Chapter 2 introduces the topic of DoS attacks and includes a review of the state-of-the-art of LDoS attack concepts, countermeasures and tools. The chapter describes and analyzes each attack with its respective countermeasure making an comparison among the approaches used by the related papers with the same attack type. In addition, attack tools were also analyzed and compared based on the traffic approaches used to disrupt the target's QoS.

Chapter 3 provides an analysis of the RoQ attack, detailing the format of the attack and how it disrupts the QoS of the targeted services. Besides, due to the lack of attack tools designed for this purpose, a new tool has been proposed and implemented. For detecting these RoQ attacks, a combination of two artificial intelligence methods and a distance metric was utilized, showing better detection results compared to other methods. All this process was made in two environments: a software emulated environment and a real environment which was implemented using the network devices of the university. This chapter concludes with a summary of the analysis and a discussion of the results and distribution parameters.

Chapter 4 provides an analysis of the Slowloris attack, outlining the attack format and its impact on the targeted services. The attack tool employed is `slowhttptest`, chosen due to its prevalence in such attacks. For detecting these Slowloris attacks, a combination of two artificial intelligence methods and a distance metric was utilized, resulting in superior detection performance for the Slowloris attack compared to alternative methods in most scenarios. The analysis was conducted across four testbed environments, including one software emulated environment and three real-world environments implemented within the Palmas city domain in Brazil. Subsequently, a summary of the analysis and a discussion of results and distribution parameters are presented.

Chapter 5 presents conclusions, including the contributions of this thesis and discusses directions for future research work.

Chapter 2

Detection and Mitigation of Low-rate Denial-of-Service Attacks: A Survey[1]

The potential for being the target of Denial of Service (DoS) attacks is one of the most severe security threats on the Internet. Attackers have been modifying their attack format over the years, damaging specific conditions of operating systems and protocols in an attempt to deny or diminish the quality of the service provided to legitimate users. Nowadays, attacks are stealthier and mimic legitimate user traffic in such a way that detection mechanisms against High-rate DoS attacks are no longer sufficient. This evolving type of attack, known as LDoS attacks, has the potential to produce more damage than its predecessor due to its stealth nature and the lack of suitable detection and defense methods. This survey summarizes and complements previous studies and surveys related to this specific type of attack. First, we propose a taxonomy of the LDoS attacks, which were divided into three broad categories based on their modus operandi: Quality of Services (QoS) attacks, Slow rate attacks, and Service queue attacks. Next, we detail numerous detection mechanisms and counter-measures available against eight types of LDoS attacks. More specifically, we describe the methods used to throttle the attack traffic. Finally, we provide a feature comparison table for some existing attack tools. This survey aims at providing an extensive review of the literature for helping researchers and network administrators find up-to-date knowledge on LDoS attacks.

2.1 Introduction

The services offered today by the Internet are of various types and features, ranging from a simple exchange of messages via e-mail to video streaming and bill payments. These services are of great importance to persons, companies, and governments, providing convenience and speed for contacting a customer, watching a movie, or requesting the delivery of products or food. However, such services can become a major inconvenience to vendors if they do not work properly for even a few minutes, since an unavailable service may cause damage to revenues and the trust of customers. A major problem with Internet protocols is that they were not originally designed to include security. Therefore, sev-

[1]The content of this chapter consists of the paper published in the following venue [29] with minor modifications: Vinícius de Miranda Rios, Damien Magoni, Pedro R. M. Inácio, Mário M. Freire, "Detection and Mitigation of Low-Rate Denial-of-Service Attacks: A Survey", IEEE Access, vol. 10, pp. 76648-76668, 2022. DOI: 10.1109/ACCESS.2022.3191430.

eral threats related to them have emerged over the last few decades. Among the existing threats against security systems are Denial-of-Service attacks, which since their inception have caused major monetary losses for people, companies, and governments that rely on the Web as their main source of revenue [38, 39]. High-rate DoS attacks have become a significant threat to Internet safety by adding the many-to-one dimension to the DoS (one computer and one Internet connection is used to swamp a target) problem [40] and amplifying it into a lethal traffic force. Distributed Denial of Service (DDoS) attacks have the aim of making resources or services unavailable to legitimate users by using a distributed, cooperative, and large-scale attack that causes damages to the system server, which may shut systems down, corrupt files, and partially or even completely compromise services [40, 41, 42]. The attack begins with the attacker, also called the botmaster or botherder (whose motivation behind DDoS attacks is mainly political/ideological, financial gain, or competitive advantage) infecting vulnerable computers (second victims) also called ”zombie machines” (generally recruited through worms, Trojan horses, or backdoors), which form a roBot Networks (Botnet) (networks formed by malware-compromised machines through the Internet), launching thousands of network traffic to the victim [43, 44, 45] in order to ruin their service, profits, and reputation. For example, from 2000 until now some of the largest and best-known companies on the Internet and entertainment (e.g. Yahoo.com, Buy.com, eBay, CNN, Amazon.com, etc) and many smaller ones have suffered DoS attacks with a huge amount of distributed network traffic that has paralysed their services [39, 46].

However, the key characteristics of DDoS flooding attacks are now well-known, and countermeasures that protect or at least attempt to protect victims from them can already be found in the literature. Thus, attackers have begun to change their strategy, denying service to the victims in a different way. They mimic legitimate user network traffic patterns by using low many-to-one dimension rate attacks, which are being called the LDoS attacks. Therefore, almost all defense mechanisms against DDoS strikes are not effective against this new attack format [22, 24].

LDoS attacks exploit specific vulnerabilities by focusing on semantic methods, but with the main feature of sending an amount of data traffic not exceeding the victim’s bandwidth. The attack signature sends a high speed burst flow, repeated at a fixed low-timescale frequency in a square wave pattern in order to cheat the system whose network links are being occupied [1, 47, 48]. For example, in 2001, Internet2 Abilene backbone was hit by short bursts of attack traffic rather than a large amount of attack traffic [49]. Asa Networks discovered that pulsing zombies are causing this type of attack. Later in 2004, the website qq.com in China was hit by some kind of Low-rate DoS attack [22]. Therefore, distinguishing LDoS attacks is a difficult task and requires solutions that can be at the same time scalable, accurate, and effective, allowing the legitimate traffic user to suffer less impact (false positive) or, in a different way, by not allowing malicious traffic to hit the victim (false negative).

The remainder of this article is divided as follows. Section 2 presents the previous surveys on LDoS. Section 3 describes a group division and concepts regarding LDoS attacks. Section 4 describes the detection methods and compares them to each other. Section 5 shows the tools used to launch the LDoS attacks traffic against the target and the tools used to defending against those attacks, while Section 6 concludes the survey.

2.2 Review of Prior Surveys on LDoS Attacks

This section presents an overview of the prior surveys targeting exclusively low-rate DoS attacks, with the aim of comparing them by their content regarding attack taxonomy, attack classification, and defense mechanisms.

In 2011, Zhu et al. provided a cursory and quick introduction about LDoS attacks [22]. They showed how shrew, Reduction of Quality (RoQ) and Pulsing DoS attacks work and demonstrate through simulations how such attacks perform in a general way.

At the same time, a comparison of attack detection mechanisms was made by Mathew and Katkar between two types of LDoS, namely Low-Rate DoS attacks against Application Servers (LoRDAS) and Shrew attacks, describing how such mechanisms act by avoiding a succession of attacks as well as pointing out the characteristics related to the features that an ideal solution must have [23].

In 2012, a brief comparison between flood-based DoS attacks and LDoS attacks was made by Liu et al. in [24]. They compared both DoS concepts and showed that new defense mechanisms needed to be created in order to stop LDoS attacks due to their stealthy mode of operation.

In the same year, Mohan et al. in [25] produced a survey of the evolution proposed by Mathew and Katkar in [23], adding other detection mechanisms, of which some are based on the AQM Random Early Detection (RED) algorithm and its variations. The authors also presented a defense mechanism based on a modified version of Random Early Detection-Preferential Dropping (RED-PD) that detected LDoS attacks, but did not specify which LDoS type of attack had been detected, even though the architecture of the proposal demonstrated that the attack was probably a Shrew attack.

A taxonomy of Slow DoS attacks was proposed by Cambiaso et al. in 2012, classifying them based on the attack characteristics, covering only malicious activities that target the application layer [26]. One interesting fact was that they considered the LoRDAS attack as a web threat that had the same slow DoS attack characteristics, which in other related papers were classified as a low-rate attack.

The following year, Cambiaso et al. in [21] described the evolution of the taxonomy since their previous paper in [26]. In the latter, they added new attacks and divided them into 3 new divisions based on the attack type and feasibility. They also adjusted the LoRDAS

attack with an appropriate classification that was related to target an application running in a server not being exclusively a web application.

In early 2020, a survey on low-rate DoS attacks written by Zhijun et al. presented the most important active LDoS attacks, with an extensive overview of these malicious activities [20]. They classified and categorized them based on their characteristics. They also expanded the discussion about the detection and defense mechanisms used by researchers, comparing their methods and techniques in order to mitigate the attacks. However, the papers studied in this survey date back to 2018 and earlier. Since then, tens of papers on LDoS attacks have been published.

In 2021, a research study presented by Tripathi and Hubballi in [27] classified the attacks based on Application Layer Protocols (ALP). This survey does not specifically target LDoS attacks but covers ALP-only attacks. They divided them into two categories: protocol specific and generic. The protocol specific category summarizes the application protocols that have a specific vulnerability to be hit. The generic category summarizes the application protocols that have generic vulnerabilities to be hit. In all the categories the authors presented a comparison among the attacks as well as the defense mechanisms.

In the same year, a survey [28] and several papers focusing on regular DoS attacks [50, 51, 52, 53, 54] were published. Our survey aims to close the gap concerning recently published papers focusing on LDoS attacks. Indeed, the aforementioned studies about LDoS attacks need to be updated with new information on issues such as new attacks, new defense mechanisms, and new classifications due to this ever-growing attack paradigm. Therefore, in order to stop or at least mitigate them, it is vital that researchers and network managers recognize and envision the LDoS attacks as a whole. Thus, one of the main contributions of this paper is to provide a state-of-the-art perspective, as well as an extensive and comprehensive high-level guide to LDoS attacks, specifically isolating and emphasizing the characteristics related to these malicious activities.

Table 2.1 provides a macro perspective on the major surveys and related papers in this area, comparing them in several important aspects related to Low-rate DoS attacks. The *Network security attack contextualization* field provides papers that have a detailed description of the LDoS. The *Attack type classification* field provides papers which have classified the threats according to certain characteristics. The *Attack tools* field provides papers that explain the tools built for each threat to be successful in its task. The *Attack target* layer field is divided into 3 categories, which are: *Application layer*, *Transport layer* and *Network layer*. It includes papers which have attacks that damage the target in each one of these categories. The *Attack performance* field shows the accuracy result for each detection/defense mechanism against the LDoS attacks. The *Low-rate DoS attacks* field is divided into 3 categories: *QoS attacks*, these include attacks that aim at reducing the throughput of data traffic transmission; *Slow DoS attacks*, which include attacks that have the aim of stopping the target service, consuming its resources and damaging the protocols of the application layer, and the *Service queue attacks*, which include attacks

Table 2.1: Comparison of prior surveys and this article.

Works	Network security attack contextualization	Attack type classification	Attack tools	Attack target			Attack performance	Low-rate DoS attacks			Detection mechanisms
				Application layer	Transport layer	Network layer		QoS attacks	Slow DoS attacks	Service queue attacks	
Zhu et al. (2011) [22]	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
Mathew and Katkar (2011) [23]	✓	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓
Liu et al. (2012) [24]	✓	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓
Mohan et al. (2012) [25]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
Cambiaso et al. (2012) [26]	✓	✓	✗	✓	✓	✗	✗	✗	✓	✓	✓
Cambiaso et al. (2013) [21]	✓	✓	✗	✓	✓	✗	✗	✓	✓	✓	✗
Zhijun et al. (2020) [20]	✓	✓	✗	✓	✓	✓	✗	✓	✗	✓	✓
Tripathi and Hubballi (2021) [27]	✓	✓	✓	✓	✗	✗	✗	✗	✓	✗	✓
This survey (2022)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

that have the aim of degrading the performance of the applications processing by damaging the processor queuing of the incoming packets. And finally, the *Detection mechanism field*, which includes papers that explain the methods and techniques for mitigating LDoS attacks.

2.3 Low-rate DoS Attacks

This section presents the Low-rate DoS attack taxonomy, schematized in Figure 2.1, which is categorized into 3 division types, based on the damage caused to the protocol used by the services, that are: QoS attacks, Slow DoS attacks and Service queue attacks as described below.

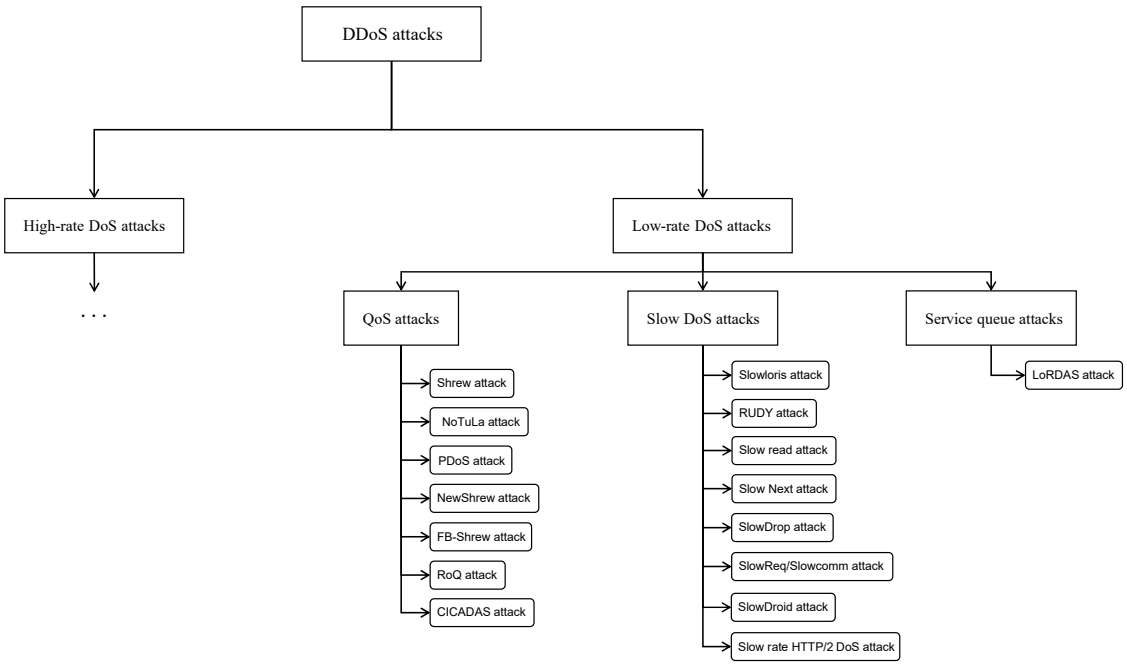

Figure 2.1: Low-rate DoS attacks taxonomy.

2.3.1 QoS Attacks

The attacks in this category have the aim of damaging the quality of services provided by applications hitting the protocols at the transport and network layers. In this case, the protocols that needed the confirmation statement are the ones most affected by these types of attacks. Figure 2.2 shows the traffic model used as basis for generating the attack traffic pattern.

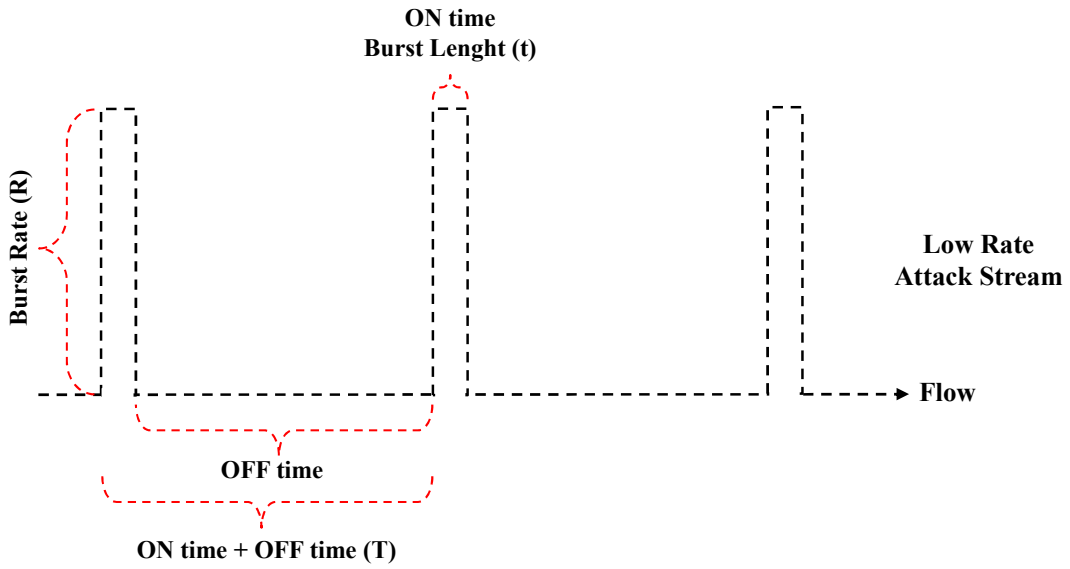

Figure 2.2: RoQ attack format (based on [1, 2, 3]). Where t is burst length of the traffic, R is the burst rate of the traffic and T is the total time for the attack period.

2.3.1.1 Shrew Attack

This attack has the aim of reducing the traffic throughput of the TCP-based application services to nearly zero. The attacker throttles the legitimate TCP flows with low-rate periodic on-off "square-wave" packets, using multiple distributed or single sources, synchronizing the attack period with the TCP minimum RTO, making the target system service consecutively repeat frequency states of overburden with a fixed frequency, shutting off most of the legitimate TCP sources [1, 55, 56, 57].

2.3.1.2 NoTuLA (Novel Tuneable Low-Intensity Adversarial Attack)

This attack has the aim of reducing the QoS of the victim's services by sending tunable traffic bursts at tunable periodic times in order to strangle the throughput for the victim [58]. The difference between that and the Shrew attack is that Novel Tuneable Low-Intensity Adversarial Attack (NoTuLA) tunes the attack traffic based on 2 stages that are: the monitoring phase and link capacity estimation phase. Instead of sending traffic in a fixed burst periodic time transmission, it is sent in a tuned burst (large enough to create transient congestion) with tuned periodic time (adjusting the inter-arrival time) transmission, with moments of silence when needed.

2.3.1.3 PDoS (Pulsing Denial-of-Service) Attack

Such an attack also aims at degrading the QoS of the victim's services, but targeting two TCP characteristics [59]. The first focuses on the default target of most of low-rate attacks, which is the retransmission timeout that the authors called timeout-based attack. On

the other hand, the second target focuses on the congestion window (cwnd), which was called aimd-based attack. The former attack forces the victim to enter the fast recovery state endlessly. It is important to mention that both attacks are made in the synchronous and asynchronous mode. The main differences between the Shrew attack and the Pulsing Denial-of-Service (PDoS) attack are the aimd-based attack and the asynchronous attack mode.

2.3.1.4 NewShrew Attack

This attack has the aim of degrading the QoS of the victim's services by targeting the retransmission timeout and slow start mechanism [60]. The first target is the default for TCP-based attacks. The second one has the aim of disrupting the TCP throughput in order to send burst traffic at the moment that the TCP is recovering from the timeout phase and entering the slow start phase, which quickly begins regaining its transmission rate. Next, the union of the TCP traffic with little attack traffic, forces the TCP to enter in the timeout stage again. The main difference between the NewShrew attack and the Shrew attack is the slow start target.

2.3.1.5 Full-Buffer Shrew Attack

This attack was first idealized by Guirguis et al. in [61] and subsequently improved by Yue et al. in [62] and [63]. The aim is to send high-rate traffic bursts only after the queue router buffer is full. This tactic produces maximum damage with minimum resources. The main difference between the Full-Buffer Shrew attack and the Shrew attack is that the first one does not need to match the minimum TCP RTO.

2.3.1.6 RoQ Attack

This attack has the aim of reducing service quality to legitimate users. The attacker throttles the network traffic in order to end systems, transmitting high-rate bursts on longer timescales, and flooding the border router queue on which most legitimate user packets are dropped [2, 30, 64, 65]. The attack target, which in this case can be any transport layer protocol, is what differentiates the Shrew from the RoQ attacks.

2.3.1.7 CICADAS Attack

This attack has the aim of degrading the QoS of the victim's services to legitimate users in synchronized coordination. The bots scattered around the internet send low-rate periodic on-off "square-wave" packets towards the victim. Due to the fact that there is no central controller of the bots, the attack traffic synchronization is made by the feedback-based

algorithm, adjusting the phase and magnitude of each attack stream based on previous Round Trip Time (RTT) measurements in a dynamic manner [66].

2.3.2 Slow DoS Attacks

The attacks in this category ”slow down” the HyperText Transfer Protocol (HTTP) traffic connections, manipulating the methods and characteristics of the protocol architecture, hence, keeping the network channel active for as much time as it can, consuming resources from the server. In 2009 Iran government servers were hit by this type of attack [26].

2.3.2.1 Slowloris Attack

This attack has the aim of stopping web server services by opening hundreds of connections and keeping them open as long as it can.

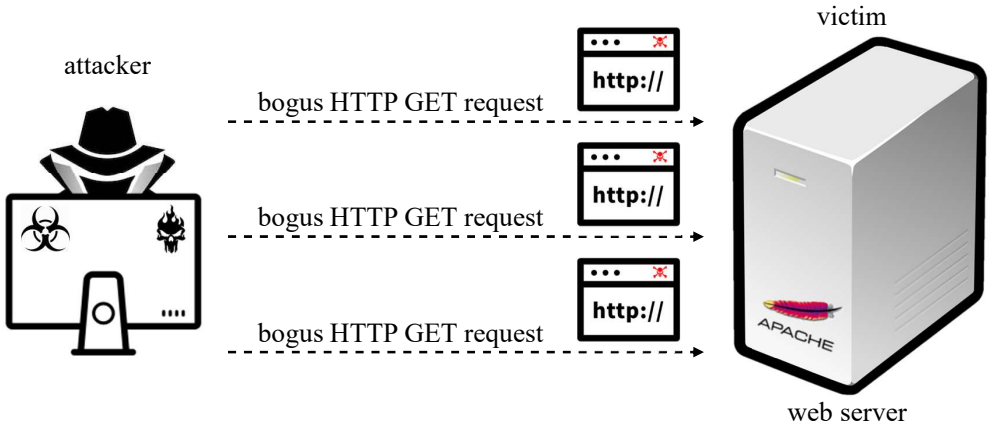

Figure 2.3: Slowloris attack.

The attacker sends partial HTTP GET requests (adding additional $\backslash r \backslash n$ tags at the end of the request) periodically and simultaneously, with different source ports to the web server opening several other connections, causing it to wait until each of the attack packet requests is completed [26],[67]. Figure 2.3 illustrates this attack representation. Once it has filled the targeted concurrent connection at its maximum, all additional connection attempts are denied.

2.3.2.2 RUDY (R-U-Dead-Yet) Attack

This attack is also called slow HTTP post attack and it has the aim to send fake HTTP POST requests disguised as the legitimate form submission, with the content length header field set as abnormally long. Additionally, the data is broken into packets as small as 1 byte each, which are sent at randomized 10-second intervals that keep the web server tied up endlessly [68], [69]. Figure 2.4 illustrates this attack.

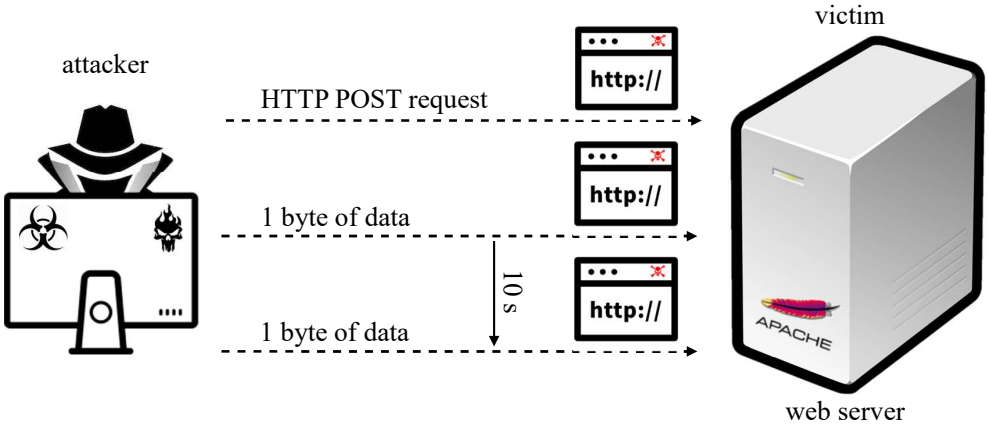

Figure 2.4: Rudy attack.

The length of the long content field forces server connections to stay open, causing them to crash.

2.3.2.3 Slow Read Attack

This attack explores the flow control of TCP by slowly reading the response with the window size smaller than usual [70], [71].

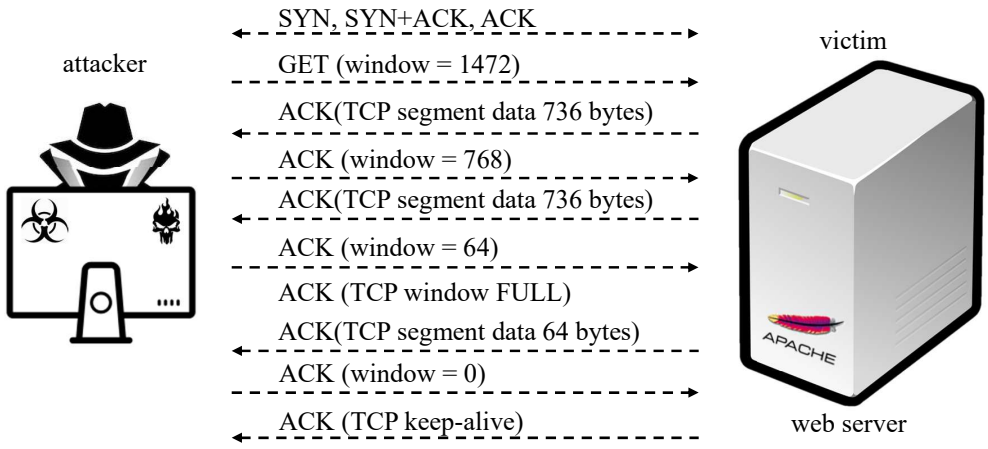

Figure 2.5: Slow read attack (based on [4]).

As a result, the attacker forces the web server to operate several connections simultaneously, which consume the resources until it can no longer receive any further requests. Figure 2.5 illustrates the attack. This malicious activity is also known as the *low and slow attack* [72].

2.3.2.4 Slow Next Attack

This attack explores the persistent connections of the HTTP protocol, targeting the timeout connection in order to keep messages alive [73]. It sends a considerable amount of traffic with bogus timeout connection values higher than usual, keeping the web server channel open and maintaining the resource busy endlessly. As a result, legitimate connections are dropped.

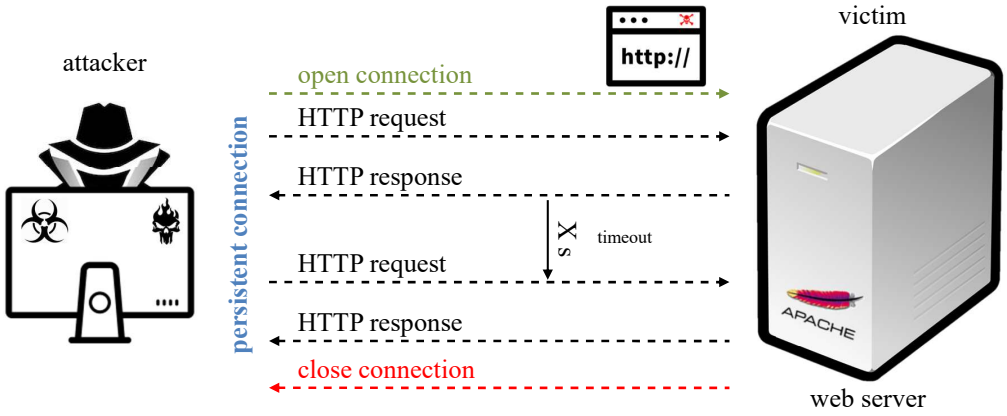

Figure 2.6: Slow Next attack.

Where X is the amount of bogus waiting time in seconds used to keep the channel open. Figure 2.6 illustrates this attack.

2.3.2.5 SlowDrop Attack

This attack opens several connections towards the web server, dropping the HTTP responses.

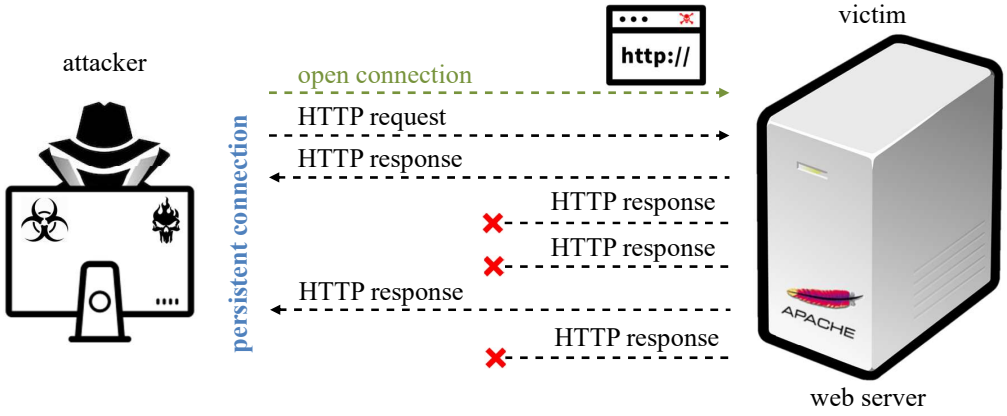

Figure 2.7: SlowDrop attack (based on [5]).

The aim is to simulate an environment where responses packets can be easily lost, such as a weak wireless connection, hence obliging the web server to endlessly respond to the

attacker [5], causing the legitimate connections to drop. Figure 2.7 illustrates this attack.

2.3.2.6 SlowReq/Slowcomm Attack

This attack is a type mix between the Slowloris attack and RUDY attack and has the aim of disrupting web server services. The attacker initially sends partial HTTP GET requests (with no $\backslash r \backslash n$ tags at the end of the request) towards the victim and after that begins to send small packets (e.g. a single character) as a form of keeping the connections alive, denying the services to the legitimate users [74], [75].

2.3.2.7 SlowDroid Attack

This attack has the same attack characteristics as the SlowReq/Slowcomm attack except that it runs on an android phone [76].

2.3.2.8 HTTP/2 DoS Attack

This attack can render a web server useless by using specially crafted HTTP/2 requests [77]. In the first crafted HTTP/2 request, the malicious client sends a modified HTTP header with the settings_initial_window_size field configured to zero, which indicates that the client cannot receive any data at that moment. This starts a waiting time by the server for the window_update frame for a specific set of time. In the second crafted HTTP/2 request the malicious client sets and resets the end_headers and end_stream fields inside a complete HTTP POST method to the web server. This causes a waiting time by the server for one or more data frames that have not yet been received, resulting in a particular time duration. In the third crafted HTTP/2 request, the malicious client sends a connection preface to the web server and thus begins waiting for a GET/-POST HTTP request which never arrives, causing a particular time duration. In the fourth crafted HTTP/2 request, the malicious client sends two types of incomplete bogus HTTP header methods. The GET and POST methods reset the end_headers and end_stream fields which indicate to the web server that there are other frames coming, causing a particular time duration. In the fifth crafted HTTP/2 requests the malicious client sends complete HTTP GET/POST requests in that the web server answers with a data frame along with two settings frame in that the second settings frame needs to be acknowledged by the client. If this does not happen the web server waits for a specific period of time. This particular time duration for each type of crafted HTTP/2 requests can be unlimited for some web servers, denying the services for the legitimate users or forcing them to wait for an extended period of time before beginning to process the requests again. That causes a long wait to access the service, leading the legitimate clients to give up.

Table 2.2: Comparison among LDoS attacks.

Attack	Target	Goal	Can IP be spoofed ?	Attack software	Previous knowledge
Shrew attack (2003) in [39]	TCP minimum RTO	Decrease QoS of the applications	Yes	No	TCP retransmission timeout
RoQ attack (2004) in [30]	Router queue	Decrease QoS of the applications	Yes	No	Border router
NoTuLa attack (2005) in [58]	TCP minimum RTO	Decrease QoS of the applications	Yes	No	TCP retransmission timeout value
PDoS attack (2005) in [59]	TCP minimum RTO and AIMD mechanism	Decrease QoS of the applications	Yes	No	TCP retransmission and congestion window
FB-Shrew attack (2006) in [61]	Router queue	Decrease QoS of the applications	Yes	No	Border router
NewShrew attack (2014) in [60]	TCP minimum RTO and slow start mechanism	Decrease QoS of the applications	Yes	No	TCP retransmission timeout and slow start initial phase
CICADAS attack (2016) in [66]	TCP minimum RTO	Decrease QoS of the applications	Yes	No	Previous RTT measurements
Slowloris attack (2009) in [26]	GET method	Stop web server services	No	Yes	HTTP header field
Slow read attack (2012) in [70]	TCP window field	Stop web server services	No	Yes	TCP window size
SlowReq/Slowcomm attack (2014) in [74]	GET method + content length field	Stop web server services	No	No	HTTP header field
SlowDroid attack (2014) in [76]	GET method + content length field	Stop web server services	No	No	HTTP header field
Slow Next attack (2015) in [73]	Wait timeout	Stop web server services	No	No	HTTP persistent connections
RUDY attack (2016) in [68]	POST method + content length field	Stop web server services	No	Yes	HTTP header field
HTTP/2 DoS attack (2018) in [77]	GET/POST methods + settings_initial_window_size field, window_update field, connection preface, end_headers field, end_stream field and settings field	Stop web server services	No	No	HTTP header field
SlowDrop attack (2019) in [5]	HTTP response	Stop web server services	No	No	webserver resources
LoRDAS attack (2007) in [6]	Data processing queue	Decrease QoS or deny the applications services	Yes	No	Queue inter-output time

2.3.3 Service Queue Attacks - LoRDAS Attack

The attacks in this category directly affect the data processing queue of the services provided by applications in the Internet. The LoRDAS attack has the aim of not allowing legitimate incoming packets to be processed by the application services for the legitimate users at the moment of response.

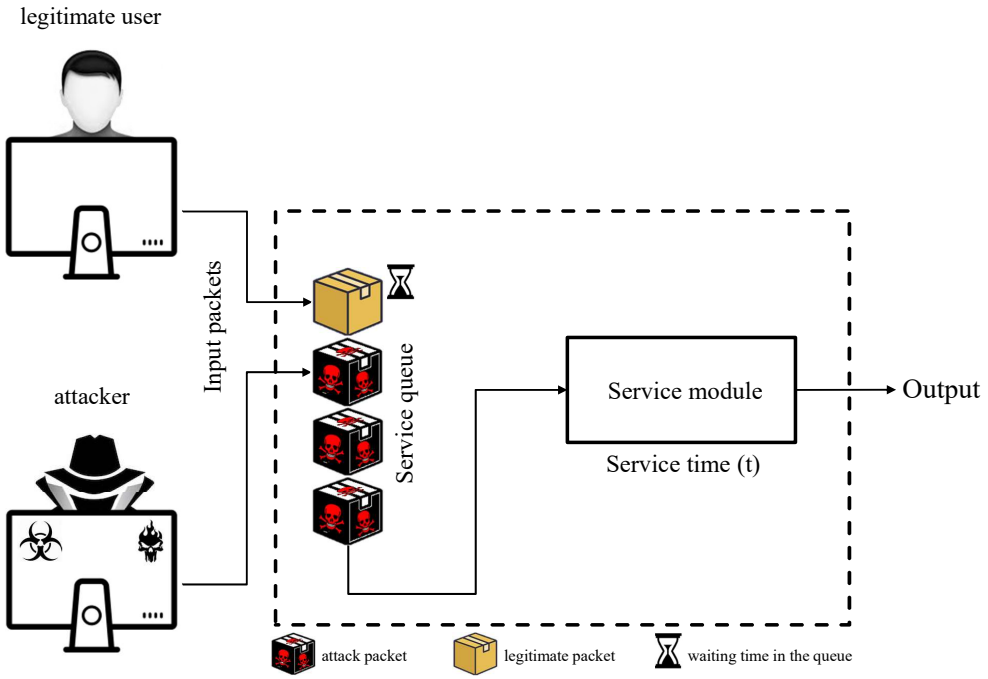

Figure 2.8: LoRDAS attack (based on [6] and [7]).

The attacker causes the service queues of the server to be overloaded and watches to see whenever responses to requests received for a particular service occur, in order to repeatedly insert a malicious request into the service queue. This way, the server is induced by the attacker to be occupied most of the time serving its requests instead of legitimate users [6, 7, 78]. Figure 2.8 illustrates the attack.

Table 2.2 shows a concise overview of the attacks presented in this section where the *Attack* field indicates the name of the menace. The *Target* field informs what the attack targets are in order to achieve success with its goal. The *Goal* field informs what the purpose of the attack is. The *Can the IP be spoofed?* field tells whether it is possible to spoof the attacker's IP address. The *Attack software* field informs if there is a software to launch the attack and the *Previous knowledge* field includes information on previous knowledge in order for the attack to be successful.

2.4 Detection and Defense Mechanisms

This section presents the detection mechanisms and countermeasures against the attacks presented in the previous section. Tables 2.3, 2.5, 2.6 and 2.7 provide a brief comparison among the strategies for stopping/defending against the threats. These Tables are compounded by the Work field, which represents the papers that produce the technology, the Detection/Defense mechanism field which represents the methodology for fighting against the attacks, the Detection/Defense location which represents the place where the methodology acts, the Testbed field which represents how the experiments were put in practice, and the Performance field which represents the performance of the methodology in detecting and/or defending against LDoS attacks.

2.4.1 Shrew Attack

This section was divided into 5 parts, which are: simulated, emulated, real environment, traffic traces and multiple testbeds. Part 1 is related to research that simulates software such as ns-2, ns-3 etc as the testbed. Part 2 is related to research that uses emulated software like netkit, Common Open Research Emulator (CORE) etc as the testbed. Part 3 is related to research that uses real environments such as a lab., university infrastructure etc. Part 4 is related to research that uses traffic traces collected in the internet as the basis for the traffic classification. Part 5 is related to research that uses multiple testbeds.

2.4.1.1 Simulated

Guang et. al. in [79] decided to randomize the TCP RTO with the aim of eliminating the future prediction of the synchronized next round of the timeout value by the attacker. They specified 3 value ranges to randomize the retransmission timeout, which worked very well in preventing the attack from learning the next RTO value as well as preserving TCP fairness against the shrew attack.

In order to detect shrew attacks Haibin et. al. in [80] chose to install detection mechanisms in the routers that were 1 hop away from the victim. The routers sniff the input port searching for anomalies in the network traffic using the Dynamic Time Wrapping (DTW) method for that purpose. This method works by comparing the similarity among the signatures of the attack with the input signal extracted from the network traffic. Once the attack is confirmed, the attacked router sends back the detection to upstream routers connected to the input port, trying to detect the attack as close to its malicious source as possible. The results showed that the detection mechanism is accurate and robust in detecting the attack.

Yu-Kwong et. al. in [81] chose to detect shrew attacks in the bottleneck link router, using a method that they called Halting Anomaly with Weighted Choking (HAWK). This method

used a flow table that maintained the statistics about the flows which were marked as ”potentially malicious” or ”confirmed malicious”. Each incoming packet was checked and if it matched some flow in the table entry its statistics were updated. After that, the router's queue was analyzed and if it was larger in size than the maximum threshold the incoming packet was dropped, but if the size was within the threshold averages (both minimum and maximum), then the packet was admitted with a P probability, otherwise the packet was truly admitted. The results showed that HAWK outperformed other router AQMs and provided an adequate experience for the legitimate user traffic.

Detecting shrew attacks is not an easy task; therefore Yu et. al. in [82] developed an algorithm by analyzing the frequency-domain characteristics, aiming at filtering, identifying, and detecting the attacks, thus preventing the legitimate traffic from being dropped. For this task, the authors considered the number of packets caught by the signal, sampling them in every 1 ms and then converting the time-domain series into their frequency domain representation using Discrete Fourier Transform (DFT). After that the Normalized Cumulative Amplitude Spectrums (NCAS) of TCP and shrew flows were compared. A flow was considered legitimate if the value of NCAS was greater than the threshold, otherwise the flow was considered as an attack. The results showed a high accuracy in detecting the attack.

Amey et. al. in [83] based their detection mechanism against shrew attacks in the RTT (Round Trip Time) traffic. The detection module was hosted in the edge routers and computed the RTT average high and low of the flows in both directions. Additionally, the RTO period was computed and estimated. If the average RTT high repeated periodically and the RTO estimated was second, then the traffic collected was malicious. The results were satisfactory and the module was easily implantable.

Shrew attacks can be defeated by increasing the buffer size of the queue routers. Sandeep and Andreas in [84] concluded this using a mathematical model that calculates the amount of the packets in the router's queue based on the burst traffic arriving to the victim. The results showed that the attacks can be mitigated by increasing the router buffer.

Amey et. al. in [85] showed through experiments that the quality of voice services was compromised by Shrew and RoQ attacks, eliminating the premise that these attacks were only effective against applications that used TCP protocol, since Voice Over Internet Protocol (VOIP) applications used UDP traffic. Based on that they decided to use the Forward Error Correction (FEC) model with Reed-Solomon code to recover the VOIP traffic from packet loss. The results showed that the FEC with RS(3,2) was effective in detecting the anomalies inside the legitimate traffic.

Shrew attacks remain as a potential strike against costumer's services. So, Zenghui and Liguo in [86] suggested to detect this attack based on a threshold, which was composed by the average of the percentage of the attack divided by the legitimate packets in the router's queue. The results were satisfactory in detecting the attack.

Changwang et. al. in [87] proposed a structure that filtered and detected Shrew attack packets passing through the route, which they called Robust RED (RRED). The idea was to detect the attack packets according to the time between arrivals, based on the dropped packets using RED algorithms. A packet arriving at the server queue would be considered suspicious if it arrived within a brief time interval after a packet coming from the same flow was dropped by the Detection and Filter block or after a packet belonging to any flow that was dropped by the RED block. The results showed that the tool was able to improve the TCP traffic under shrew attack.

In order to mitigate the impacts of shrew attacks against the target, Huaping et. al. in [88] made a comparison between two queue managements, RED and Droptail. They used both in the same scenario conditions. The results showed that the greater the distribution of the attack nodes, the greater was the effectiveness of droptail defense and the worse the effectiveness of the RED.

Kumawat and Meena in [89] created a framework for detecting LDoS attacks. First of all the traffic was characterized as malicious or normal through the Entropy attribute. If the Entropy was greater than the threshold, then the traffic would be considered malicious, otherwise, the traffic would be regarded as normal. Next, in order to detect the type of the attack the traffic entropy was compared to a threshold. If the Entropy was high it would be marked as a High-rate attack; if the Entropy was low it would be marked as a Low-rate attack, otherwise it would be marked as normal. Finally, the attack was mitigated based on the flow ID of the traffic. The results were successful in detecting the attacks.

Detecting shrew attacks in a Mobile Ad hoc Network (MANET) is a difficult task. With that in mind, Singh et al. in [90] proposed a mechanism that set the congestion bit based on three status values, these being frequency of receiving Request to Send (RTS)/Clear to Send (CTS) packets, frequency of sensing a busy channel, and the number of RTS/DATA retransmissions, which were observed by a passive server. It verified whether the three status values exceeded the limit values set by the threshold. If that were true, the traffic would be marked as an attack and all the traffic would be blocked or rejected. The results showed a reduction in both the attack and the packet loss of legitimate users.

A detection mechanism called Spectral Energy Distribution Probability (SEDP) was created by Wu et al. in [91] to detect shrew attacks. The idea behind the approach was to work with the signal generated by the amount of attack and legitimate traffic. The authors treated a shrew attack as a short signal and the TCP traffic as a long signal. Based on that it was necessary to change the time domain to a frequency domain using Fourier transform. Next, the spectral energy of legitimate and attack traffic was calculated in such a way that if the energy exceeded the threshold the alarm would be triggered. The results showed better accuracy in detection and less consumption in computation.

An extended approach based on Xiang et al. in [92] was made by Sahoo et al. in [93], using the union of the Generalized Entropy (GE) and Generalized Information Divergence (GID)

metrics. In order to compute the flow entropy it was verified whether the packet window sizes of the traffic flow were equal to 80; if that was the case, then the destination IP and occurrence were computed in a hash table. This action also computed the information distance among the flows. After computing the GE and GID from the flows, the values generated were compared to the set thresholds. If the values were found to be greater than or equal to the threshold, then the counter parameter increased by 1. If the counter parameter was equal to 5, then the flow was marked as an attack. The results made it possible to identify the attack traffic and the legitimate traffic with an improved false negative rate.

Zhang et al. in [94] proposed detecting the shrew attack based on the extension of the PCA algorithm called adaptive Kernel Principal Component Analysis (KPCA). First, all the sample data were extracted from the wavelet multi-scale analysis and then the Radial Basis Function (RBF) parameters, number of main components, and Squared Prediction Error (SPE) trust limit were trained and adjusted regarding the network environment. If the values of the parameter exceeded a certain threshold, then an attack alarm would be triggered. The results showed a 99.2% accuracy with a 0.8% false negative rate and 2% false positive rate.

Liu et al. in [95] created a detection mechanism that classifies the network traffic based on frequency-domain, grouping them in clusters based on NCAS features. The proposal was to separate the sampled traffic in clusters, using the BIRCH algorithm for this and then blocking the clusters with NCAS values above the threshold. The results showed an accuracy beyond 70% and a rapid response time.

A defense/detection mechanism called Fair Robust Random Early Detection (FRRED) algorithm was developed by Lin et al. in [96]. The proposal for detecting the attack was to record the packet drop time, checking if the incoming packet belonged to the same flow of the packet dropped and if it arrived in a short-range time after the dropped one. Based on that, the indicator labeled as f.I was used to classify the flow as ”under attack” or ”legitimate”, which meant that if the verified packet was under attack, then the f.I would decrease by one, otherwise it would be increased by one. The results showed an effective throughput and fairness for TCP flows and mitigation for attack flow traffic. Huang et al. in [97] used the Cross-Correlation Identity Distinction (CCID) method with the aim of identifying the stealth Shrew attack traffic mixed with the legitimate traffic. For this task the authors converted different data flows into the appropriate time domain signal and frequency domain. After that the cross-correlation of both domain type was calculated. Next, the traffic with high cross-correlation coefficient was marked as an attack traffic. The results showed that in both domain type the CCID was effective in detecting the attack.

In order to detect and filter Shrew attacks, Şimşek and Şentürk in [98] based their study on the congestion queue and standard deviation of the traffic in the router. For detection, the main difference from other AQM-based researches was that they based it on the pre-congestion period of the router queue. They assumed that the attack would begin after a

packet was dropped. In order to filter the attack traffic they used the standard deviation method for the variation in time of the packets arrival. The results showed an efficient detection with no false positive and negative rates.

In 2018, Siracusano et al. [99] have proposed a methodology for the detection of LDDoS attacks based on the characteristics of malicious TCP flows. They have used two datasets: one generated from a simulated network, the other from the publically available CIC DoS dataset. Both contain the attacks slowread, slowheaders and slowbody, alongside legitimate web browsing. They have extracted TCP flow features from all connections and have shown that decision trees and K-NN supervised algorithms accurately classified up to 99.99% of flows. In the same year, Cotae et Rabie [100] proposed a game theoretic approach to detect LDoS attacks. They simulated network congestion attacks and created a threshold bandwidth filter at the router that allows a specific bandwidth. They considered the game players in a static simultaneous game and found the Nash Equilibrium where players do not have any profit. They concluded that a mixed strategy will the best response for an organization using this approach.

In 2021, Liu et al. [101] have developed a novel semi-supervised Locality Sensitive-Incremental Transductive Support Vector Machine (LS-ITSVM) method. Their method incorporates local frequency-domain features from the autocorrelation sequence of network flows into the regularization time-domain framework of Transductive Support Vector Machine (TSVM). Simulation results show an higher detection accuracy of abnormal network flows, a faster training and better response times.

2.4.1.2 Emulated

The detection approach against Shrew attack by Kaur and Agrawal in [102] involved two phases. The first was to generate a log file with the traffic network (attack + legitimate) as the input data. The second was to apply the detection attack into the log file aiming at analyzing how accurate the suggested method was. The chosen method was based on a modified version of the Changepoint Detection Algorithm (CDA), which they called Quickest Changepoint Detection (QCD) algorithm. The algorithm has the aim of observing any small changes in the traffic probability deviating from the previous normal behavior and locating in real time the exact moment where the anomaly change occurs.

2.4.1.3 Real Environment

Ying et. al. in [103] described how the Border Gateway Protocol (BGP) could be affected by shrew attacks as well as techniques that mitigated the damage generated by this attack. The first thing they did was to hide information such as the min RTO value from the BGP packet, and the second thing was to guarantee the bandwidth and to prioritize scheduling for BGP traffic using for this task a set of solutions as filters and queuing methods, e.g.,

Weighted RED (WRED). The results were effective in prevent the attacks.

In order to detect shrew attacks, Gautam et. al. in [104] made a comparison between two detection techniques called Modeled Attack Detector (MAD) and the modified Periodic Attack Detector (PAD) based on Xinming et. al. in [105]. The MAD technique operated on the sampled time-series of network traffic, whereas PAD worked in the spectrum of network traffic. The results showed that MAD was able to detect attacks faster than PAD.

Yang et. al. in [92] proposed a combination of two information metrics for shrew attack detection, generalized entropy and information distance. These metrics were compared to two known approaches, Shannon entropy and Kullback-Leibler distance. The detection process started by collecting all data traffic passing through the routers 2 steps away from the victim and then it calculated the distribution probability from all the sampled data. After that the values were sent to routers 1 step away from the victim and their distances were calculated. The DDoS attack was detected in case the distance summed was greater than the threshold. The experimental results showed an effective detection and low false positive rate.

The detection system created by Rejo and Vijay in [106] used Intrusion Detection System (IDS) software in addition to a software-based approach to detect shrew attacks. The software worked in the first moment, collecting the data traffic and calculating the average traffic data, number of timeouts and the amount of dropped packets. In the second moment, it used the values generated in the previous step in order to calculate the current and average inter-arrival time in the time slot. After that, it verified whether the values were higher than the threshold; if that was true, the traffic would be an attack. The results were shown to be effective in detecting the attacks.

In order to keep the target of the shrew attack from being overwhelmed, Wu et al. in [107] used the Principal Component Analysis (PCA) algorithm to define the Open Supervised Device Protocol matrix of a normal flow, and one under attack. The flow ID is composed of source address, source port, destination address, and destination port. Based on that, a waveform of the normal and attack traffic can be drawn to select the appropriate threshold. That way it is possible to detect the attack with good performance and achieve a high detection rate, low false alarm, and low missed alarm probability.

In order to detect shrew attacks in Software-Defined Networking (SDN) networks, Agrawal and Tapaswi in [108] created a defense scheme containing the detection of the attack through entropy and tracing back the traffic attack to its origin. The entropy mechanism was used for detecting the attack, which was able to differentiate the attacked traffic from the traffic without attack. After that, with the attack source IP addresses in hand it was possible to verify the origin of the attack using a Deterministic Packet Marking (DPM) traceback scheme, blocking the attack using Access Control List (ACL) and SDN flow-table rules. The results showed a 97.6% accuracy in detecting Shrew attacks.

Boro et al. in [109] designed a mechanism that rapidly detects the shrew attack in the

traffic inflow. For this purpose they used the Self-Similarity Matrix (SSM) across multiple time scales, computing the similarity between pairs of features, which are Average packets per network flow, Number of packets per interval or sample, Number of network flows and Server outflow performance in a set of time-ordered data samples. The results showed that the mechanism was effective in terms of accuracy and computation speed.‘

Tang et al. in [110] developed a framework called Performance and Features (PF) for real-time LDoS attack detection in SDN networks. The framework is divided into three parts, which are: Feature Extraction and Classification, Attack Detection and Attack Mitigation. The Feature Extraction and Classification module has the aim of extracting features from the perfomance of the TCP traffic under LDoS attack and from the characteristics of the LDoS attack traffic. Next, the Attack Detection module has of aim of detecting anomalies based on the features collected from the Feature Extraction and Classification module. Finally, the Attack Mitigation module will locate the source IP of attackers and the victim ports, based on the rules that it sends according to the locating result to filter LDoS attack traffic. The results showed a detection accuracy performance of 96%.

2.4.1.4 Traffic Traces

Bhuyan et al. in [111] experimented with information metric theory algorithms, namely Hartley entropy, Shannon entropy, Renyi's entropy and Generalized entropy with the aim of quantifying their success in detecting shrew attacks. Basically, the data was sampled into 10s period received by the upstream routers and then the distribution probability was calculated based on the flow ID (Source IP, Destination IP and Protocol) to generate the Entropy. And finally the value obtained was checked in order to verify whether it was greater than the threshold. If true, the flow would be marked as an attack, otherwise the flow would go to the next router. The results were effective in detecting the attack.

Bhushan and Gupta in [112] based their detection tool on the hypothesis test defining the H_0 as $\sigma_N = \sigma_R$ which represented the legitimate traffic and H_1 as $\sigma_N > \sigma_R$ which represented the attack traffic. The edge routers would sample the traffic arriving to them and calculate the standard deviation σ_R based on the packet size of each packet and a threshold. After that the hypothesis was tested in order to verify whether the traffic was legitimate or not. The results showed effective detection by the tool.

2.4.1.5 Multiple Testbeds

Zhijun et. al. in [113] detected attacks using a comparison between the similarity of the signal from a synthetic shrew attack and the heterogeneous signal composed of the TCP flows plus the shrew attack traffic. The results of the calculation were compared to the correlation value based on a threshold, which was calculated based on experiments. If the value of the similarity overstepped the threshold, the traffic would be classified as an

Table 2.3: Comparison among detection/defense mechanisms against Shrew attacks.

Work	Used approach for traffic classification	Detection/Defense mechanism (location)	[Testbed] / Performance
Guang et al. [79]	-	RTO randomized (victim)	[Simulated] / Effective
Haibin et al. [80]	Auto-correlation plot signature of the traffic attack	Dynamic time wrapping (routers one hop away from the victim)	[Simulated] / -
Yu-Kwong et al. [81]	Average queue size of the router	Halting Anomaly with Weighted choKing (border router)	[Simulated] / HAWK outperforms other AQM and provides a fair QoS to legitimate traffic
Yu et al. [82]	Flow ID (IP source address, IP destination address and IP destination port)	Normalized cumulative amplitude spectrum (server)	[Simulated] / High detection accuracy achieved using a collaborative distributed detection mechanism
Amey et al. [83]	Flow ID (IP source address, IP source port, IP destination address and IP destination port)	Average RTT and estimated RTO of the traffic (edge router)	[Simulated] / -
Sandeep and Andreas [84]	-	Increase buffer size of the routers (border router)	[Simulated] / -
Ying et al. [103]	-	Hide BGP topology information and prioritize routing traffic (border router)	[Real] / Effective
Gautam et al. [104]	Sampled time-series of network traffic	Modeled attack detector (-)	[Real] / -
Amey et al. [85]	-	FEC model with Reed-Solomon code (-)	[Simulated] / The FEC model with RS(3,2) is effective in detect the anomalies inside the legitimate traffic
Zenghui and Liguo [86]	-	The rate of the dropping packets (border router)	[Simulated] / -
Changwang et al. [87]	Flow ID (IP source address, IP source port, IP destination address, IP destination port and Protocol)	RRED (border router)	[Simulated] / Is highly robust, can greatly improve the performance of TCP under attacks, outperforming existing RED-like algorithms
Huaping et al. [88]	-	Droptail and RED (border router)	[Simulated] / The deeper degree of distribution, the better defense performance of Droptail (passive queue management) and the worse defense performance of RED
Yang et al. [92]	IP characteristics	(border router and routers 1 and 2 hops away from the victim) Generalized entropy and information distance metric	[Real] / Effective detection and low false positive rate
Rejo and Vijay [106]	Traffic inter-arrival time and dropped packets	Software-based approach integrated with existing IDS (detection server)	[Real] / -
Zhijun et al. [113]	The correlation between the signal of simulated shrew attack and the hybrid signal (TCP flow plus shrew attack)	Spectral intervals (border router)	[Real and simulated] / False negative alarm rate < 1.1%; False positive alarm rate < 0.8%; The detect rate > 98%
Kai et al. [114]	Variation of TCP traffic	Exponential Weighted Move Average (border router)	[Simulated and traffic traces] / Effective detection with low false positive rate
Changwang et al. [115]	Congestion router queue and drop packets	Congestion participation rate (border router)	[Simulated, real environment and traffic traces] / Effective
Kumawat and Meena [89]	Entropy	Low false positive rate (intermediate routers)	[Simulated] / Effective
Singh et al. [90]	Frequency of receiving RTS/CTS packets, frequency of sensing a busy channel and the number of RTS/DATA retransmissions	Status values threshold (Passive server)	[Simulated] / Reduce attack throughput there by increasing the received bandwidth and reducing the packet loss of legitimate users
Wu et al. [107]	Flow ID (IP source address, IP source port, IP destination address and IP destination port)	PCA algorithm (Firewall)	[Real environment] / High detection rate, low false alarm and low missed alarm probability
Bhuyan et al. [111]	Flow ID (IP source address, IP destination address and IP destination port)	Entropy methods (-)	[Traffic traces] / Effective
Wu et al. [91]	Network traffic	Spectral Energy Distribution Probability (Victim)	[Simulated / The detection accuracy is higher with low false positive and negative rate
Wu et al. [116]	Holder parameter	Multifractal analyzes (Victim)	[Real environment and simulated] / Simulation achieved 92% of accuracy and False positive rate of 9% while the testbed environment achieved 91% of accuracy and False positive rate of 10% and low complexity

Table 2.4: Comparison among detection/defense mechanisms against Shrew attacks (continued).

Work	Used approach for traffic classification	Detection/Defense mechanism (location)	[Testbed] / Performance
Sahoo et al. [93]	Destination IP and occurrance	Generalized Entropy and Generalized Information divergence metrics (Controller)	[Simulated] / Improved false negative rate
Zhang et al. [94]	Network traffic	Kernel Principal Component Analysis (-)	[Simulated] / 99.2% of accuracy with 0.8% of false negative rate and 2% of false positive rate
Kaur and Agrawal [102]	Network traffic	Change point Detection Algorithm (-)	[Emulated] / -
Agrawal and Tapaswi [108]	IP source and IP packet size	Entropy and IP traceback mechanism (SDN Controller)	[Real environment] / 97.6% detection accuracy
Boro et al. [109]	Network traffic	Self-Similarity Matrix (Firewall)	[Real environment] / Effective
Liu et al. [95]	Frequency-domain	Network flow grouping method + NCAS (Border router)	[Simulated] / Detection accuracy greater than 70%
Lin et al. [96]	Flow ID (IP source address, IP source port, IP destination address, IP destination port and Protocol)	FRRED (Border router)	[Simulated] / Effectively preserve the throughput of TCP flows Significantly improve fairness among TCP flows; Effectively mitigate address-spoofing LDoS attacks
Huang et al. [97]	Frequency domain and Time Domain Signal	Cross Correlation Identity Distinction (Victim)	[Simulated] / Effective
Şimşek and Şentürk [98]	Router queue and delay time	Precongestion period and arrival times of packets (Border router)	[Simulated] / Very low false positive and false negative rates
Bhushan and Gupta [112]	Packet size	Hypothesis test (Edge routers)	[Traffic traces] / Effective
Tang et al. [117]	Variance and mean deviation parameters	SADBSCAN algorithm and cosine similarity (Victim)	[Simulated, real environment and traffic traces] / Improve the detection accuracy and reduce the false negative rates
Tang et al. [110]	Time-frequency analysis, packet transfer speed, average packet transfer speed, average traffic speed, packet variance, packet standard deviation, coefficient of variation of packet and average packet size	P&F framework (Target switch)	[Real environment] / 96% of detection accuracy

attack. The results had high accuracy and performance.

A Shrew attack is an anomaly traffic, which can be defeated. The authors Kai et. al. in [114] showed that the attack can be detected based on the abnormal phenomena of network traffic. First, all the traffic data is sampled and next, the Exponential Weighted Move Average (EWMA) algorithm generated statistics about the hybrid traffic. Second, the detection method observes the behavior of the traffic distribution generated by EWMA algorithm and makes various judgment criteria. If all criteria match in the time window then the attack will be detected. The results were effective in detecting attacks on the traffic.

Changwang et. al. in [115] proposed a measurement defined as Congestion Participation Rate (CPR) for detecting and filtering shrew attacks. The measurement was based on the proportional relationship between congested incoming packets versus the total incoming packets in the stream. If in this proportional relationship there was a CPR greater than the predefined threshold, then the flow would be considered an attack and all packets would be dropped. The results were shown to be effective against the attacks.

Wu et al. in [116] proposed an approach based on a multi-fractal analysis of network traffic. Protocols such TCP, IP and HTTP have this characteristic, while UDP is a monofractal. The approach needed to set the Hölder exponent parameter in order to detect the attack. Therefore, the smaller the exponent α was, the more confident would be the result that the traffic was malicious. The results based on the simulation achieved 92% of accuracy and a false positive rate of 9% while the testbed environment achieved 91% of accuracy and a false positive rate of 10%.

Tang et al. in [117] proposed a Low-rate DoS attack detection mechanism that adaptively identifies traffic attacks in clusters of data in multidensity datasets. To achieve this goal the authors created an algorithm called Self-Adaptive Density-Based Spatial Clustering of Applications with Noise (SADBSCAN) which has the aim of improving noise and data distribution as well as mitigating the effects of data class imbalance produced by classical clustering algorithms. To detect the LDoS attack the data traffic is first split into equal amounts of multiple data units where in each unit the variance and mean deviation parameters are extracted from the TCP and UDP traffic to be used as eingevalues into the SADBSCAN algorithm. Next, the cosine similarity is introduced to label the clusters and noise points resulting from SADBSCAN algorithm. This label could be attack traffic or benign traffic based on the value threshold provided. The results showed that the detection mechanism improves accuracy in distinguishing Low-rate DoS attacks from legitimate traffic, which reduces the false negative rates.

2.4.2 RoQ Attack

The detection mechanism provided by Arunmozhi and Venkataramani in [64] was an integrated scheme in that the intermediate nodes kept on sending an Flow Monitoring Ta-

ble (FMT) file to the destination mode, which contained the flow ID, source ID, packet sending rate, and destination ID. If a node was experiencing a congestion, it would send a packet to the destination node with the congestion experienced (CE) bit marked in the IP header. That way, the destination node that received the flow with the CE bit marked would send a control warning to the source asking it to diminish the sending rate. If the source node did not stop the flow, then it would be considered an attacker and all the flow belonging to it would be discarded.

Guirguis et al. in [65] studied the impact of the RoQ attacks on admission controllers and load balancing servers. They observed that the attack could be very powerful in disrupting the efficiency of the servers. Based on that, an admission ratio β parameter value has been inserted in order to be dynamically adaptive, which was based on the equation:

$$\alpha^n(i) = \frac{1}{N} + \beta \sum_{j=1}^{N} (q^j(i-a) - q^n(i-1)) \quad \textbf{(2.1)}$$

Setting the β value to 0.03 the load balancers might display a quicker reaction to the changes in traffic, keeping services available. The admission controllers' K parameter has the same effect.

Ren et al. in [118] created a defense scheme with the aim of avoiding a decrease in quality of data related to voice and video traffic in a MANET. This scheme was divided into 2 parts. The first part acted in the MAC layer by keeping track of the frequency and retransmission of the packet flow. After that, the monitored values were analyzed and if they exceeded a certain threshold an alert would be triggered. The second part of the defense scheme was indicating the packets with the congestion bit field and warning the sender nodes about the congestion problem. If they did not reduce their transmission data, they would then be considered as attacking nodes. The results showed that the delay and decrease of traffic quality was proportional to the sum of attack traffic flows.

Shevtekar and Ansari in [2] proposed a detection environment system that was sectioned into two phases. The first or detection phase started with the sudden increase of the packets above a certain virtual queue threshold. Next, all the new arriving packets were analyzed to identify whether they belonged to the benign flow table. If that was true the packets would be forwarded to the attack filtering, otherwise they would go to the normal path. In the attack filtering stage, the algorithm checked how long it took for each packet to arrive in each flow, thus examining the disparity in time between each arrival. If the time difference between those arrivals was very small, the detection mechanism module would perform a packet load count, and in case the valued exceed a certain threshold, the result would be that the attack was detected. Once the attack was detected, malicious packets were dropped. The detection system for RoQ attack and Low-rate DoS attack successfully achieved its purpose. Chen and Hwang in [34] designed a detection scheme that diffe-

rentiated a normal TCP flow from a RoQ attack flow by making use of spectral analysis. The scheme used the hypothesis testing method in order to differentiate the traffic under attack from the legitimate one based on the spectrum of the flow in the frequency band. The results showed that the scheme was able to cut off RoQ attack flow and effectively save 99% of legitimate TCP flows.

Gulati and Dhaliwal in [119] began the detection system by choosing the computer selected to be the attack monitor. After the selection, if the elected computer node verified that, within a small period of time above the threshold there was a rapid increase in the amount of traffic, the flow would be placed on a suspect list. If any suspect node appeared a large number of times, it would be added to a checking table list. This table list was later sent to every node within the network environment whose amount of traffic passing through was above a certain threshold throughout a specific time period. If this was true, then the node would be placed in the attacker table; otherwise, it would be dropped from the suspect table list. Each node listed in the ”attacker table list” was to be blocked. The results showed that it was viable to diminish packet loss and to have better throughput rates.

Wen et al. in [120] created a detection system divided into two stages. the first stage was to analyze suddenly anomalies in the collected traffic data. Next, the sampled traffic was sliced by time and checked by the Cumulative Sum (CUSUM) method. If any slice showed any anomaly, then the second stage was removed. At this point an auto-correlation analysis was used to verify the signal periodicity. If it was consistent along with time, an alarm would be triggered confirming that the RoQ attack was present. The results showed an accurate detection scheme with low false positive and false negative rates.

Gang et al. in [121] tested the network traffic performance of the Mobile Internet Protocols such as Mobile Internet Protocol version 4 (MIPv4), Mobile Internet Protocol version 6 (MIPv6) and Fast Mobile Internet Protocol version 6 (FMIPv6) against the RoQ attack. In this scenario the proposed detection solution used the Hamilton-path-based scheme as the main factor to decrease the packet loss of the legitimate traffic nodes. The aim was to create a reserved bandwidth in the overlay networks to allow the legitimate traffic to be forwarded to the destiny. As the results showed, without the detection solution the mobile protocols suffered more damage due to the amount of traffic passing throw them. On the other hand, with the detection scheme, the packet loss and delay decreased in the overlay networks.

Hongsong et al. in [122] explored three methods (Ensemble Empirical Mode Decomposition (EEMD), Hilbert-Huang Transform (HHT)) in order to create a detection mechanism that could eliminate issues that were likely to be related to the signal generated by the RoQ attack, such as mode mixing and fake components as well as analyzing and comparing the time–frequency distribution of routing message traffic. Hence, based on the Intrinsic Mode Function (IMF) spectrum of the traffic generated by the detection scheme, it was easy to detect the attack. The results showed a very accurate detection of the RoQ

Table 2.5: Comparison among detection/defense mechanisms against RoQ attacks.

Work	Used approach for traffic classification	Detection/Defense mechanism	Detection/Defense location	Testbed	Performance
Arunmozhi and Venkataramani in [64]	-	Flow Monitoring	Destination node	Simulated	Achieves higher throughput and packet delivery ratio
Guirguis et al. in [65]	-	In the admission controller's equation Dynamically adapt beta and k values	-	Simulated	-
Ren et al. in [118]	-	Based on the threshold of the frequency of receiving RTS/CTS packets, frequency of sensing a busy channel, and the number of RTS/DATA retransmissions	-	Simulated	-
Shevtekar and Ansari in [2]	Flow ID (packet count, packet size, createtime and lastaccessed time)	Router-based approach	Routers close to the victim	Simulated	Can successfully detect and mitigate RoQ attacks
Chen and Hwang in [34]	Flow ID (IP source address, IP source port, IP destination address and IP destination port)	Flow-level spectral analysis with sequential hypothesis testing	-	Simulated	Effectively rescue 99% of legitimate TCP flows
Gulati and Dhaliwal in [119]	-	Flow Monitoring Table	Chosen attack monitor	Simulated	Reduce packet loss and improves throughput
Wen et al. in [120]	Abruptly traffic changes	CUSUM method with auto-correlation analysis	-	Simulated and Real environment	High accuracy and high Effective
Gang et al. in [121]	-	Hamilton-path-based scheme	-	Simulated	Reduced packet loss
Hongsong et al. in [122]	Intrinsic Mode Function spectrum	Hilbert-Huang Transform with ensemble empirical mode decomposition and Correlated coefficient method	-	Simulated	Highly effective
Rios et al. in [37]	Flow ID (IP source address, IP source port, IP destination address, and IP destination port)	Fuzzy logic, Neural Network and Euclidean distance	Victim	Emulated and Real environment	100% of accuracy in the emulated environment and 99.3% of accuracy in the real environment
Liu et al. in [123]	Average values of the attack and background traffic bands and energy intensity of the attack flow sub-band	Self-adjusting SVM method with APSO algorithm	Victim	Emulated	Higher detection accuracy, which ranges from 92.36% to 96.65%

attack traffic.

In order to detect RoQ attacks, Rios et al. in [37] proposed a detection mechanism which was composed of fuzzy logic, neural network and Euclidean distance. All these methods used a training data set as the basis for classifying the data as attack or legitimate. The aim was to classify the data using fuzzy logic and neural networks and then to compare the classification generated by both. If the same data values had different classifications, the Euclidean distance was used to check them with the training data set classification. After that, the result of each classification method was compared and if there were more attack classifications than legitimate ones, the data would be classified as an attack, otherwise it would be classified as legitimate. However, if there was a tie, the data would be classified as a warning that would require a closer inspection by the network administrators. The results showed an excellent performance with 100% of accuracy in the emulated environment and 99.3% of accuracy in the real environment in detecting the attack traffic.

Liu et al. in [123] created a Low-rate detection mechanism which was compounded by the union of the self-adjusting Support Vector Machine (SVM) method with the Adaptive Particle Swarm Optimization (APSO) algorithm. The self-adjusting SVM method has the aim of enhancing the generalization ability of the pure SVM algorithm and it was made using 2 approaches, which are: the adjustment of the gamma parameter and the adoption of the optimal degree value from the training data. The APSO algorithm was used to enhance the adjustment of the attack and background traffic band average values and the energy intensity of the attack flow sub-band parameters. The results showed excellent detection performance with the accuracy ranging from 92.36% to 96.65%.

2.4.3 Slowloris Attack

Aiello et al. in [124] proposed a Statistical Based Intrusion Detection (SBID) approach, which had the aim of detecting slowloris attacks based on the probability distribution of the parameters Δrequest, Δresponse, Δdelay and Δnext for two distinct traffic $f(x)$ and $g(x)$ belonging to the same network situation. If the two traffics had distinct distributions, the alarm was triggered. The results showed 100% of anomaly detection, with 1% probability of false positive rate.

In order to detect and soften the application DoS attack slowloris, Dantas et al. in [125] proposed a defense mechanism that they called Selective Defense for Application Layer DDoS Attacks (SeVen). It assumed state-dependent protocols and used the notion of state in the messages of the HTTP protocol. The software did not immediately respond to requests received by the application, but instead waited for a period of time called the round. During a round, the messages were accumulated in an internal buffer and when the buffer reached its limit, it was decided whether a new request would be accepted or not, based on certain criteria. It is important to notice whether the new request was accepted or not, based on the probability, since the attack packets were more likely to be dropped. The re-

sults showed high levels of availability, greater robustness, and they also led to less traffic than other techniques.

In order to detect slow DoS attacks, including the slowloris attack, Mongelli et al. in [126] proposed a detection method with the aim of analyzing specific spectral features, such as the number of received packets of traffic over small time horizons. The time horizon analyzed the current and previous temporal packets behavior in order to see significant changes in both traffic. Hence, in order to identify the attack they used the Fast Fourier Transform in the current time horizon in frequency domain. The attack was detected due to smoother behavior of the legitimate traffic.

Katkar et al. in [127] configured an IDS software along with Naive Bayesian algorithm with the aim of quickly detecting the slowloris attack. To validate the results they created a test-bed environment that was composed of web servers clusters, due to the fact that this approach has benefits for the web server application. All the traffic arriving to the web servers was collected and formatted for a traffic data file type and transmitted to be analyzed by the IDS software. It categorized the received aggregated records into normal or intrusive ones. Whenever a record was categorized as intrusive, it was an indication that the server was under DoS/DDoS attack.

Based on the HTTP traffic request and response, Aqil et al. in [128] determined the optimum detection high and low threshold based on features for detecting slowloris attacks. Basically, the detection algorithm checked every 3 seconds if there were feature values above or below the determined thresholds. If that was the case, the algorithm warned of an attack. The results were shown to be an extremely effective approach.

Hirakawa et al. in [31] based the detection scheme on 3 steps. The first was to check whether the numbers of connections that were monitored were greater than a threshold during a normal time. If the threshold was trespassed, then in step 2 all the IPs belonging to the same flow and that were appearing most frequently would be disconnected. In step 3, if the connections were above the threshold, that stopped the disconnection process and all the cycles started over. The results showed enough resistance against the strikes from a single attacker as well as from distributed strikes from 30 attackers by calibrating the threshold appropriately.

In order to detect the slowloris attack Singh and De in [129] created a scheme by uniting the genetic algorithm method, which was used to improve the Multilayer Perceptron Neural Network (MLP) Neural Network weights and the MLP Neural Network algorithm. They made a comparison between the proposal with other machine learning algorithms such as Naive Bayes, RBF Network, MLP, J48, and C45. In the tests the authors' algorithm was better at detecting the slowloris attack, reaching 98% of accuracy.

Faria et al. in [130] developed a tool which they called Slowloris Detecting Tool for WMNs (SDToW) that detected and blocked slowloris attack traffic in Wireless Mesh Networks (WMN)s. First, they analyzed the slowloris traffic behavior. After that, they based the

Table 2.6: Comparison among detection/defense mechanisms against Slowloris attacks.

Work	Used approach for traffic classification	Detection/Defense mechanism	Detection/Defense location	Testbed	Performance
Aiello et al. in [124]	Probability distributions parameters Δrequest, Δresponse, Δdelay and Δnext	Statistical Based Intrusion Detection	Victim	-	100% of anomalies detection, with 1% probability of false positive rate.
Dantas et al. in [125]	-	SeVen	Victim	Simulated	High levels of availability,greater robustness and also leads to less traffic
Mongelli et al. in [126]	-	Fast Fourier Transform	Victim	Traffic traces	Frequency domain
Katkar et al. in [127]	-	IDS with Naive Bayesian classifier	Victim	Traffic traces	IDS signature
Aqil et al. in [128]	The volume of data sent, the volume of data received, Interrupts, context switches and TCP Sockets	High and low traffic threshold	Victim	Real environment	The true positive rate is close to 100% and the false positive rate is decreased by about 66% as compared to traditional detectors
Hirakawa et al. in [31]	The number of connections and duration time for each IP address	Amount of connections per IP address	Victim	Simulated	Enough resistance and effective
Singh and De in [129]	The number of HTTP count, the number of the IP addresses, the constant mapping function and the fixed frame length	MLP-GA	Victim	-	Detection accuracy of 98.04%
Faria et al. in [130]	HTTP GET method, The Reassembled PDU and the Packets with 296 bytes and TCP set on protocol field	SDToW	Victim	Real environment	Lower incidence of false positive errors

detection on three information items, the HTTP GET method, The Reassembled Protocol Data Unit (PDU), and the Packets with 296 bytes and TCP set on protocol field. In every 5 seconds, traffic was collected by the Collection Module (CM) in the web server and it was sent to the concentrator where the Analysis and Filtering Module (AFM) selected the connections with the information listed above. If there was traffic within these three information inside the traffic connections, so there was an ongoing attack and the IP and Media Access Control (MAC) addresses of the connections would be extracted and included into the blacklist to be blocked. The results showed a lower incidence of false positive errors.

2.4.4 NewShrew Attack

Cotae et al. in [131] based their detection proposal on the Fisher g-statistics test method. In order to detect the attack they set a frequency interval in the range of 0.01Hz to 1.1Hz which was labeled as Shrew frequency attack interval. They performed a simulation of the Fisher g-statistics tests for one and multiple time series. For both time series they considered an attack if the content was in the Shrew frequency attack interval and the p-value was less than 10^{-3}. The results showed effective detection, with all attacks being identified.

Luo and Chang in [59] proposed a detection system based on two stages, having as reference the presence of two traffic anomalies. The first one was the incoming fluctuate data traffic and the second one was the low amount of outgoing ACK traffic. In the first stage it applied the discrete wavelet transform, which was used to monitor these two anomalies. In the second stage the CUSUM algorithm was applied in order to detect both anomalies. The experiments were highly accurate in detecting PDoS attack traffic.

2.4.5 PDoS Attack

The Vanguard detection system developed by Luo et al. in [132] was an extension of the Luo and Chang approach in [59]. This system employed more metrics to detect three traffic anomalies induced by the attackers using a CUSUM algorithm. The first anomaly was the incoming fluctuate data traffic, the second one was the decline of the outgoing ACK traffic, and third, the changes in the distribution of the incoming TCP data rate. The results showed an accurate detection system, capable of detecting in a short-range time a wide range of attack traffic.

2.4.6 RUDY Attack

Najafabadi et al. in [68] made a classification comparison among 3 machine learning algorithms namely K-Nearest Neighbors (K-NN) and two forms of C4.5 decision trees (C4.5D and C4.5N). The algorithm classification comparison firstly used a set of 'N' features and

Table 2.7: Comparison among detection/defense mechanisms against NewShrew attack, PDoS attack, RUDY attack, HTTP/2 DoS attack and LoRDAS attack.

Work	Used approach for traffic classification	Detection/Defense mechanism	Detection/Defense location	Testbed	Performance	Attack type
Cotae et al. in [131]	Frequency interval in the traffic range	Fisher G-statistic test	-	Simulated	Correctly identified all attacks	NewShrew
Luo and Chang in [59]	Incoming fluctuate data traffic and the amount of outgoing ACK traffic	Wavelet transform and CUSUM	Victim	Emulated	Very effective	PDoS
Luo et al. in [132]	Incoming fluctuate data traffic and amount of outgoing ACK traffic and distribution of the incoming TCP data rate	CUSUM	SNORT server	Real environment	Effective	PDoS
Najafabadi et al. in [68]	Features: Outbound session convergence, Inbound session convergence, Packets, Bytes, RIOT Bytes, Outbound velocity bpp and Outbound velocity bps	K-Nearest Neighbor and two forms of C4.5 decision trees (C4.5D and C4.5N)	Border router	Real environment	Very good classification performance	RUDY
Tripathi and Hubballi in [77]	Source IP address and source port number	Chi-square test	Victim	Real environment	Recall ranging from 8.33% to 100% for and FPR of 0% for all the scenarios	HTTP/2 DoS
Maciá-Fernández et al. in [47]	-	RST, RAI, RTQB and IRTQB	Victim	Simulated	-	LoRDAS

then a set of 7 features (selected as the best ones among the set of 'N' features). The first result showed that the features related to traffic size, self-similarity, and speed were very effective in detecting the R-U-Dead-Yet (RUDY) attack. The second result showed that in both classification results C4.5N was the best algorithm with low false positive rate among the machine learning algorithms.

2.4.7 HTTP/2 DoS Attack

Tripathi and Hubballi in [77] mitigated the HTTP/2 DoS attack using the Chi-square test. First, in the training phase the authors collected data from legitimate traffic and then in the testing phase, the current traffic was compared to collected traffic. For this comparison, the Chi-square test was used along with five features that are: settings frame having settings_initial_window_size set to 0, headers frame having end_stream_flag reset, flows having Connection Preface only, headers frame having end_headers_flag reset and Server's settings frames which were not recognized. Therefore, in a ΔT time the current traffic was collected and compared to the training phase traffic and then it was calculated through the Chi-square test. If the obtained calculated value was less than the predefined threshold, the detector did not contain anomalous flows. However, if the calculated value was higher than the previously established threshold, the detector contained anomalous flows. The results from the 7 scenarios had a recall ranging from 8.33% to 100% and a False Positive Rate (FPR) of 0%.

2.4.8 LoRDAS Attack

Maciá-Fernández et al. in [47] proposed 4 defense mechanisms against LoRDAS attack that were: Random Answer Instant (RAI)), Random Service Time (RST), Improved Random Time Queue Blocking (IRTQB) and Random Time Queue Blocking (RTQB). The RST method had the aim of randomizing the answer instant response, thus avoiding allowing the attacker to predict the correct moment to send the attack. In the RAI method the authors decoupled the instant method from the enable method, thus avoiding the attacker to event predict the instant method; the enable instant would not suffer any damage. Due to RST and RAI limitations related to time, which directly impacted the server behavior the RTQB method was developed with the aim of inserting a blocking interval between the answer instant and the enable instant, thus preventing the attacker from successfully damaging the legitimate data processing and discarding all the requests arriving on it. Due to RTQB discard process, the authors improved it (IRTQB) in a way that the attack process was selectively chosen to be discarded. The results showed an effective defense by the IRTQB method if compared to the other methods.

2.4.9 Leveraging SDN Against LDoS attacks

In recent years, several papers have leveraged the use of SDN to mitigate LDoS attacks. An SDN controller has a global view of the whole network and can aggregate very fine-grained information on specific flows. As the controller is cloud-based, it can leverage much processing power and memory volume which enables it to use centralized algorithms such as those based on machine learning. In 2019, SoftGuard [133] was proposed to defend against low-rate TCP attacks in SDN. Another framework called Q-MIND [134] was proposed that same year by Phan et al. to reduce LDoS attacks bu using reinforcement learning. The following year, Perez-Diaz et al. have used six machine learning algorithms [135] in their IDS inside an SDN simulated environment. Still in 2020, Balarezo et al. have investigated the feasibility and mitigation of LDoS shrew attacks on the control plane connections of an SDN network [136]. Several other papers related to SDN appeared in 2021. Tang et al. proposed a framework based on the Histogram-based Gradient Boosting and Finding Peaks (HGB-FP) algorithm to detect LDoS attacks and mitigate their influence in SDN in real-time [137]. Ilango et al. have designed a deep learning scheme called FFCNN to detect LDoS attacks in an Internet of Things-Software Defined Networks (IoT-SDN) environment [138]. Lui et al. proposed an LDoS attack detection method based on a Two-step Self-adjusting Support Vector Machine (TS-SVM) [123]. Their proposal was evaluated by simulation and showed an improvement over a traditional SVM approach. Finally, Li et al. used a Bat Algorithm associated with a Bat Algorithm associated with a BP Neural Network (BA-BNN) to detect LDoS attacks [139]. As the previous study, the evaluation was performed on a simulated environment using *mininet* and the Ryu controller.

2.5 Tools

2.5.1 Attack Tools

In order to launch the attack traffic towards the victim, a tool specifically developed for this purpose is necessary as can be seen in Table 2.8. Some of the malicious activities presented here have a software which is open source and can be changed for the most varied purposes. The QoS attack and Service queue attacks presented in Section 3 did not have a software that was specific to the aim of the attack. Usually a Constant Bit Rate (CBR) traffic was used to meet the LDoS attack requirements as depicted in Figure 2.2. In the simulated environments, most of the papers (if not all) used CBR traffic burst as the attack traffic. In the real environment, most authors did not mention what kind of software they were using and a few others used some stress tool in a CBR traffic burst manner for the attack traffic. One exception was the paper by Rios et al. in [37] in which the authors created a script that manipulated the Hping3 tool output flood traffic to meet the

RoQ attack requirements. On the other hand the Slow DoS attack had some threats with software developed to fill its DoS requirements. The slowloris attack was the threat with most software developed for damaging the web server services followed by Slow Read, RUDY, SlowReq and Slowcomm, that were: slowloris.pl [140], PyLoris [141], QSlowloris [142], an unnamed PHP version [143, 144], SlowHTTPTest [67], Http bog [145], Torshammer [146], Goloris [147], SlowDroid [76], slowloris.py [148], R-U-Dead-Yet [149], Cyphon [150], sloww [151], dotloris [152] and pwnloris [153]. Table 2.8 describes the attack tools in detail where the *Year* field indicates the birth of the defense tool. The *Tool name* field indicates the name of the tool. The *Attack mode* field indicates the format of the attack, which are: single or distributed. The *Operating System* field indicates which system is used by the tool. The *Programming language* field indicates which language the defense tool was developed. The *Interface* field indicates which interface type is used to configure the parameters of the defense tool, if it is a terminal type or a graphical type. The *Attack type* field indicates the type o attack which the attack tool is launching.

Table 2.8: Comparison among Slow DoS attacks tools.

Year	Tool name	Operating System	Programming language	Interface	Attack type
2009	slowloris.pl in [140]	Linux	Perl	CLI	Slowloris
2009	Pyloris in [141]	Windows/Mac/Linux	Python	CLI/GUI	Slowloris
2009	QSlowloris in [142]	Windows		CLI	Slowloris
2009	unnamed PHP version in [143]		PHP	CLI	Slowloris
2011	Slowhttptest in [67]	Windows/Mac	Python	CLI/GUI	Slowloris Slow Read
2011	Http bog in [145]	Windows	C#	CLI	Slow Read
2012	Torshammer in [146]	Linux	Python	CLI	RUDY
2014	Goloris in [147]	Mac/Linux	Go	CLI	Slowloris
2014	SlowDroid in [76]	Android		GUI	SlowReq Slowcomm
2015	slowloris.py in [148]	Mac/Linux	Python	CLI	Slowloris
2016	R-U-Dead-Yet in [149]	Linux	Python	CLI	RUDY
2018	Cyphon in [150]	Mac	Perl	GUI	Slowloris
2018	sloww in [151]	Linux	JavaScript	CLI	Slowloris
2018	dotloris in [152]	Windows	C#	CLI	Slowloris
2018	pwnloris in [153]	Windows/Linux	Python	CLI	Slowloris

2.5.2 Defense Tools

In order to create a barrier to prevent that malicious traffic arrive to the target's attack, some existing defense tools were used for this purpose as can be seen in Table 2.9. The IDS (Intrusion Detection System) tools Snort and Suricata [154] were adapted to be used for DDoS attacks detection and more recently used for slowloris attack detection as in [130] and [155]. The recent SDToW tool, developed for the slowloris attack detection for mobile networks. As the IDS tools, the iptables firewall was adapted for slowloris attack

detection setting rules accordingly [156]. Table 2.9 describes the defense tools in detail where the *Year* field indicates the birth of the defense tool. The *Tool name* field indicates the name of the tool. The *Defense attack mode* field indicates the format of the attack, which are: single or distributed. The *Operating System* field indicates which system is used by the tools. The *Programming language* field indicates which language the defense tool was developed. The *Interface* field indicates which interface type is used to configure the parameters of the defense tool, if it is a terminal type or a graphical type. The *Defense against field* indicates the type o attack which the defense tool is stopping.

Table 2.9: Comparison among Slow DoS attacks defense tools.

Year	Tool name	Defense attack mode		Operating System			Programming language	Interface		Defense against
		LDoS	LDDoS	Windows	Mac OS	Linux		CLI	GUI	
1998	Snort in [130] and [155]		✓			✓	C	✓		Slowloris
1998	Iptables in [156]	✓				✓	C	✓		Slowloris
2010	Suricata in [155]		✓			✓	C and Rust	✓		Slowloris
2020	SDToW in [130]		✓			✓	Python	✓		Slowloris

2.6 Conclusion

LDoS attacks, although still at an early stage, are receiving increased attention due to the fact that most of them are not detected by High-rate DoS attack detection mechanisms. Such an attack method has been widely adopted by attackers, since it makes them stealthier and more capable of shutting down their targets. In this chapter, we presented a comprehensive study guide of the LDoS attacks, defense/detection mechanisms and tools. First, we divided the attacks into three categories for the purpose of making them more comprehensible so as to be distinguished in terms of format, target, and the purpose of the attack. Second, we presented the mechanisms that can identify the attacks based on the fingerprints left over from the attack traffic and then stop or mitigate them. Third, we showed a list of attack tools that can reproduce some of the LDoS attack concepts.

Chapter 3

Detection of Reduction-of-Quality DDoS Attacks Using Fuzzy Logic and Machine Learning Algorithms[2]

DDoS attacks are still among the most dangerous attacks on the Internet. With the enhancement of methods for detecting and mitigating these attacks, crackers have improved their skills in creating new DDoS attack types with the aim of mimicking normal traffic behaviour therefore becoming silently powerful. Among these advanced DDoS attack types, the so-called low-rate DoS attacks aim at keeping a low level of network traffic. In this chapter, we study one of these techniques, called RoQ attack. To investigate the detection of this type of attack, we evaluate and compare the use of eleven machine learning algorithms: MLP with backpropagation, K-NN, SVM, Multinomial Naive Bayes (MNB), Gaussian Naive Bayes (GNB), Logistic Regression (LR), Decision Tree (DT), ADABOOST, Gradient Boosting (XGB), Random Forest (RF) and Light Gradient Boosting Machine (LGBM). We also propose an approach for detecting this kind of attack based on three methods: Fuzzy Logic (FL), MLP and Euclidean Distance (ED). We evaluate and compare the approach based on FL, MLP and ED to the above machine learning algorithms using both emulated and real traffic traces. We show that among the eleven Machine Learning algorithms, the best classification results are obtained with MLP, which, for emulated traffic, leads to a F1-score of 98.04% for attack traffic and 99.30% for legitimate traffic, reaching an accuracy of 96.91%, while, for real traffic, it leads to a F1-score of 99.87% for attack traffic and 99.95% for legitimate traffic, reaching an accuracy of 99.96%. Regarding the approach using FL, MLP and ED, for classification of emulated traffic, we obtained a F1-score of 98.80% for attack traffic and 99.60% for legitimate traffic, leading to an accuracy of 99.36%, while, for real traffic, we obtained a F1-score of 100% for attack traffic and 100% for legitimate traffic, leading to an accuracy of 100%. However, the better performance of the approach based on FL, MLP and ED is obtained with a slightly higher cost execution time than the other machine learning algorithms to classify the emulated and real traffic datasets, respectively.

[2]The content of this chapter consists of the paper published in the following venue [37] with several modifications: Vinícius de Miranda Rios, Damien Magoni, Pedro R. M. Inácio, Mário M. Freire, "Detection of Reduction of Quality DDoS Attacks Using Fuzzy Logic and Machine Learning Algorithms", Computer Networks, Volume 186, 2021, Article 107792, Elsevier DOI: 10.1016/j.comnet.2020.107792.

3.1 Introduction

Internet services and online applications have been the target of numerous types of attacks over the years. Among them, DoS / DDoS attacks may stop or degrade the services provided to customers, sometimes causing serious financial costs and reputation damage for many online businesses [157, 158, 159, 160, 161, 162]. DoS attacks had the initial goal of consuming resources from a system or application, sending a huge amount of poorly formatted data traffic from one single computer towards the victims [39, 40, 41, 163]. A classic example of this type of attack is the Ping of Death [164]. With the rise of effective defense techniques against this type of attack, the attackers have redefined it into DDoS which is capable of blocking the resources of the target machine such as the connection link [165], applications [77, 129] and/or hardware resources [166]. DDoS attacks aim at making the old one-to-one attack into a new and more elaborate many-to-one attack, recruiting as many infected computers as possible and creating a botnet (robot network) in order to send a huge amount of data traffic towards the target machine. Some examples of this type of attack include SYN flood attack, Smurf attack, UDP flood attack, Domain Name System (DNS) flood attack (see e.g., [40, 42, 167]).

In a DDoS attack, the attacker conducts the entire process via handler computers, which through a secure channel such as Internet Relay Chat (IRC) manages the zombie computers that generate the traffic towards the victim in order to compromise its services. Despite DoS/DDoS attacks being a plague to Internet for more than two decades, in 2009 several massive DDoS attacks were carried out in order to disrupt network services of popular websites such as Facebook, Live Journal, Twitter, and Amazon [168]. Since then, the topic of DDoS has been an intense field of research, where new approaches for detection, mitigation, prevention or defense have been proposed, following the evolution of DDoS attack mechanisms (see e.g. surveys [40, 167, 168, 169, 170, 171, 172, 173]) and/or new application domains such as mobile/wireless [174, 175], Internet of Things [176], SDN [177, 178], cloud [171, 172] or fog computing [179, 180]. Nevertheless, DDoS attacks have continued to increase and to be successful causing service interruptions that lead to huge financial losses, as was the case of the Dyn attack which took place on October 21, 2016 and was initiated by a DNS operator bringing down major websites including PayPal, Amazon, Airbnb, Visa, The New York Times, Netflix, GitHub, Reddit, Twitter, Spotify, the Guardian, and CNN [181, 182, 183]. Other recent high impact attacks include the attack to Dream Host (Hosting Provider) on August 24, 2017 [184], the attack on the UK National Lottery on September 30, 2017 [184], the attack on the Electroneum cryptocurrency startup, just before it launched its mobile mining app on November 2, 2017 [184], the GitHub attack in 2018, which remains the largest DDoS attack of all times, reaching heights when it started at a rate of 1.3Tbps and sent packets at a rate of 126.9 million PPS [183], or the recent attack on the Wikipedia website on September 6, 2019, that paralyzed its site in Europe and some parts of the Middle East [185]. According to a recent report from Neustar Research, major DDoS attacks increased 967% in the first quarter of 2019

compared to the first quarter of 2018 [186], and according to the latest figures of Kaspersky, the number of detected DDoS attacks jumped 18% year-on-year in the second quarter [187].

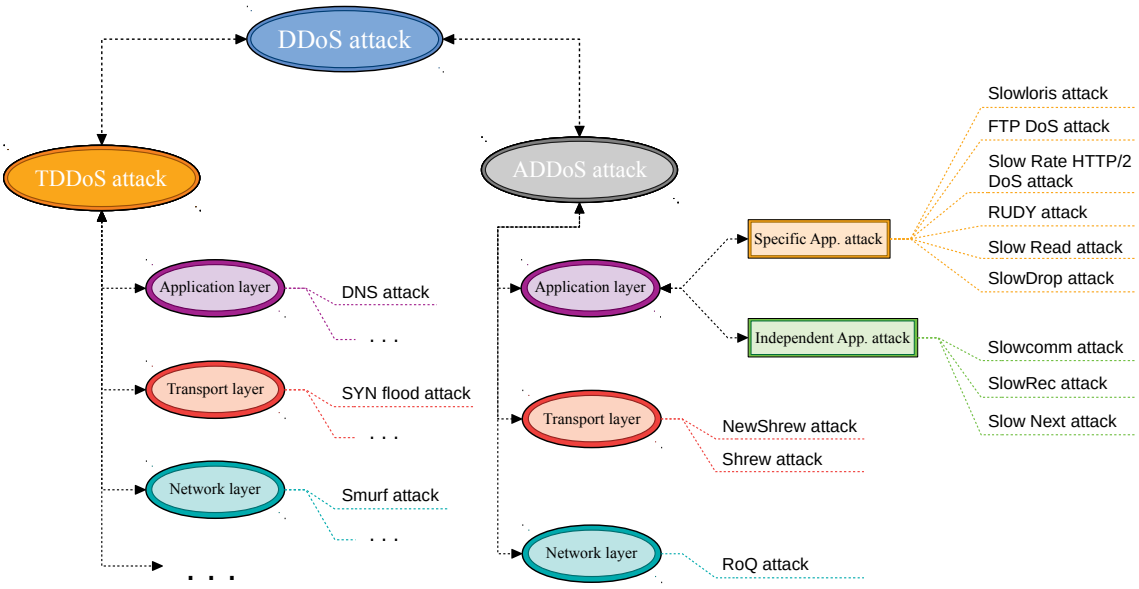

Figure 3.1: Classification of DDoS attacks.

In order to escape detection, a new kind of DDoS attacks has emerged with the aim of mimicking the legitimate traffic behaviour of users of the service. This kind of attack may not cause a service shutdown, but may decrease the quality of the offered service. As a result, DDoS attacks may be classified into two attack types: Traditional Distributed Denial of Service (TDDoS) attacks, which include the classic DDoS attacks, and Advanced Distributed Denial of Service (ADDoS) attacks, which include the new stealthy generation of DDoS attacks. Advanced DDoS attacks may be classified as low-rate DoS/DDoS (also known as LDoS) attacks or as slow DoS/DDoS attacks. Low-rate DoS/DDoS attacks include RoQ attacks [30, 120, 121] at the network layer, Shrew attack [1, 115, 188] and the New-Shrew attack [60] at the transport layer, and LoRDAS attack [6, 78] at the application layer. Slow DoS/DDoS attacks are confined to the application layer and include the following application-specific attacks: Slowloris (Slow HTTP GET) [189, 190, 191], File Transfer Protocol (FTP) DoS attacks [45], Slow Rate HTTP/2 DoS attacks [77], Slow HTTP POST also known as RUDY [189], Slow Read attack [192, 193], and SlowDrop attack [5]. These attacks have the same format characteristics, but explore different vulnerabilities. Recently, a few DoS attacks have been reported with the characteristic of being independent of the target application layer protocol, which include SlowReq [45, 74], Slowcomm [75] and Slow Next [45, 73]. An overview of the classification of these DoS/DDoS attacks is proposed in Figure 3.1.

This chapter focus on the detection of RoQ attacks by using two different approaches based on machine learning algorithms. The first approach consists in the separate use of eleven machine learning algorithms to detect the RoQ attacks, namely MLP with back-

propagation, K-NN, SVM, MNB, GNB, LR, DT, ADABOOST, XGB, RF and LGBM. These algorithms have been widely investigated for detecting high-rate DDoS attacks, e, g. MLP used in [194, 195, 196], K-NN used in [194, 196, 197], SVM used in [195, 196, 197, 198] and MNB used in [199, 200, 201], etc. The second approach consists in the joint use of a combination of three distinct methods: FL, MLP and the ED.

The remaining part of this chapter is organized as follows. Section 3.2 explains how a RoQ attack is performed and Section 3.3 reviews the related work about detection of RoQ attacks. Section 3.4 addresses the proposed approaches for the detection of RoQ attacks. Section 3.5 describes the experimental environments used for the detection of RoQ attacks and Section 3.6 discusses the obtained results. Section 3.7 presents our conclusions.

3.2 RoQ Attack

The LDoS attack aims at reducing the quality of service provided by the target to the point where data transfer rate drops to zero. However the most important characteristic that distinguishes this kind of attack from other DDoS attacks is that, in order to achieve its goal, it does not require a huge amount of data to stop the services running in the target machine. This kind of attack simply sends a quantity of traffic equal to or slightly higher than the bandwidth of the target so that detecting mechanisms do not realize that the ongoing undetected attack causes damage to applications' connections. The RoQ attack seeks to reduce the QoS of the target services regardless of the transport protocol used by those services [30, 122].

3.3 Related Work

This section provides an overview of the use of machine learning algorithms and fuzzy logic for detection of DDoS attacks. It shows the reduced number of works devoted to the detection of advanced DDoS attacks, due to the difficulty in their detection. This section also provides a brief description and a comparative analysis of methods already published for detection of RoQ DDoS attacks.

3.3.1 Use of Machine Learning Algorithms and Fuzzy Logic for Detection of DDoS Attacks

Soft computing methods (e.g., machine learning, fuzzy logic, evolutionary computation, probabilistic deduction) have been mostly used to detect traditional high-rate DDoS attacks. Khalaf et al. survey in [168] the use of artificial intelligence and statistical approaches for detection of and defense against high-rate DDoS attacks, namely Bayesian

networks, fuzzy logic, genetic algorithms, K-NN, neural networks, software agents and support vector machines. Recently, Hosseini and Azizi in [194] have used naive Bayes, random forests, decision trees, MLP, and K-NN for detecting high-rate DDoS.Sreeram and Vuppala in [202] have used machine learning metrics and a bio-inspired bat algorithm for HTTP flood attack detection, Meti et al. in [195] have used machine learning algorithms (Support Vector Machine and Neural Network) for detection of high-rate DDoS attacks in SDNs, and Mohammed et al. have proposed in [203] a machine learning-based collaborative DDoS mitigation mechanism for SDNs.

Alrehan and Alhaidari have reviewed in [204] the use of machine learning techniques to detect DDoS attacks on Vehicular Ad hoc NETworks (VANETs), Wani et al. have reported in [198] the use of machine learning algorithms (Support Vector Machine, Naive Bayes, and Random Forest) to detect high-rate DDoS attacks on a cloud environment, Hou et al. have reported in [205] the detection of high-rate DDos attacks through NetFlow analysis using Random Forest, and Aamir and Zaidi in [197] have used K-NN, Support Vector Machine and Random Forest algorithms for high-rate DDoS classification.

Fuzzy logic has also been used for detection of high-rate DDoS attacks. Balarengadurai and Saraswathi in [206] have used fuzzy logic for detection of exhaustion attacks over IEEE 802.15.4 MAC Layer, Rodrìguez et al. have reported in [207] a dynamic DDoS mitigation scheme based on TTL field and fuzzy logic, Mondal et al. in [8] have used fuzzy logic for detection of DDoS in cloud computing environments, and Alsirhani et al. have proposed in [208] a DDoS detection system using a set of classification algorithms (Naive Bayes, Decision Tree (Entropy), Decision Tree (Gini), and Random Forest) controlled by a fuzzy logic system in Apache Spark.

To the best of our knowledge, only a few works have addressed the application of machine learning algorithms for detecting low-rate DDoS attacks. In [129], Singh and De have employed a Multilayer Perceptron with a Genetic Algorithm (MLP-GA) for the detection of Slowloris attacks and in [209] Bhuyan and Elmroth have used a generalized total variation metric to detect Shrew attacks. As far as we know, no work has been reported about the use of fuzzy logic for detection of low-rate DDoS attacks.

3.3.2 Previous Methods for Detection of RoQ Attacks

This subsection provides a brief description of published methods for detection of RoQ DDoS attacks, as well as their comparative analysis presented in Table 3.1. This table also puts in evidence how different test-bed components and communication medium characteristics are taken into account by each work and this chapter. In this table, the Test-bed Type field represents environments in which attack and detection systems are tested. The Test-bed AQM represents the techniques that align the packets arriving at the router, some techniques being more efficient than others to perform this job. The Detection Mechanism field represents methods that detect and/or respond to an attack.

Table 3.1: Comparison among related works and this article.

Works	Testbed Environment	AQM Testbed	Network Type	Mechanisms	Traffic Protocol	Attack Software	Performance
Ren et al. (2007) [118]	Simulated	Absent	Wireless	To monitor the threshold of three MAC layer signals	TCP and UDP	Absent	Effective
Guirguis et al. (2007) [210]	Simulated and Real	Absent	Ethernet	Absent	Absent	Absent	Absent
Guirguis et al. (2007) [65]	Real	Absent	Ethernet	Dynamically adapt β value in admission controller's equation	Absent	Absent	Effective
Chen and Hwang (2007) [34]	Simulated	Droptail	Ethernet	Flow-level spectral analysis with sequential hypothesis testing	TCP and UDP	Absent	Effectively rescue 99% legitimate TCP flows
Shevtekar and Ansari (2008) [2]	Simulated	RED-PD	Ethernet	Router-based approach	TCP and UDP	Absent	Absent
Chen et al. (2008) [211]	Simulated and Real	Absent	Wireless	Absent	Absent	Absent	Absent
Arunmozhi and Venkataramani (2010) [64]	Simulated	Absent	Wireless	Flow Monitoring	Absent	Absent	Achieves higher throughput and packet delivery ratio
Gulati and Dhaliwal (2013) [119]	Simulated	Absent	Wireless	Flow Monitoring Table (FMT)	Absent	Absent	Reduce packet loss and improves throughput
Wen et al. (2014) [120]	Simulated and Real	Absent	Ethernet	Wavelet multiresolution analysis method with autocorrelation analysis	TCP, ICMP and UDP	absent	High accuracy and high efficiency
Gang et al (2017) [121]	Simulated	Absent	Wireless	Hamilton-path -based scheme	Absent	Absent	Reduce evidently packet loss
Hongsong et al. (2019) [122]	Simulated	Absent	Wireless	Hilbert-Huang Transform (HHT) with Ensemble empirical mode decomposition (EEMD) and Correlated coefficient method	TCP	AOMDV with RREQ	Highly efficient
This article (2020)	Emulated and Real	Droptail	Ethernet	Machine learning algorithms, Fuzzy logic and Euclidean distance	TCP and UDP	M-RoQ	See section 3.6

The Protocol field represents the protocol used by the traffic of legitimate clients and attackers. The Attack Software field represents tools that can launch RoQ attacks. Finally, the Performance field represents how the detection mechanisms were effective to detect the attacks. As we can see, this chapter has a more complete environment than previous works for testing the whole process of the system.

In order to avoid the loss of quality of voice and video traffic in a MANET by RoQ attack, Ren et al. in [118] have created two defense mechanisms. The first one detects attacks by monitoring the frequency and the retransmission of packets in the MAC layer and the second one responds to the attack by marking the packets with the congestion bit, notifying the emitters. The obtained results show that the higher the amount of attack traffic flows, the greater the delay and the loss of the traffic quality.

To evaluate the RoQ attack against dynamic load balancers, Guirguis et al. in [210] have employed the RoQ attack power metric in order to observe the feedback delay, the resources managed and the average variance in the features in the attack moments, damaging the performance of the entire network. Their results have shown that an attack can cause serious damage against load balancers and they have shown how future defense mechanisms can be created.

Guirguis et al. in [65] have implemented the admission ratio's β parameter value to be dynamically adaptive in the equation:

$$\alpha_i^n = \frac{1}{N} + \beta \sum_{j=1}^{N} (q_{i-1}^j - q_{i-1}^n) \quad (3.1)$$

In this way, the load balancers can react faster to the changes of the traffic, keeping the services available. The same happens to the admission controllers' K parameter. The results have shown an increased efficiency against the attack. To avoid the servers to be overwhelmed by RoQ attacks, Chen and Hwang in [34] have created a novel defense mechanism by combining the flow-level spectral analysis (which can segregate normal TCP flows from malicious flows) with sequential hypothesis testing. Their results have shown that this new detection system can effectively rescue 99% of the legitimate TCP flows under RoQ attacks.

Shevtekar and Ansari in [2] have addressed the detection of the RoQ attack in two phases. The first detection phase is initialized by the sudden increase of the packets in the queue. Subsequently, the time difference between consecutive arrivals of the packets of each stream is checked; if it is too short, a packet load count is made and if this value is greater than the threshold, the attack is detected. After detection, packets marked as attack packets are discarded. Their results have shown an effective countermeasure approach for RoQ attack and Low-rate DoS attack.

A novel RoQ attack model was proposed by Chen et al. in [211] with the aim of jamming the MAC layer at a special moment, which can repeatedly block TCP ACK transition and

degrade wireless TCP throughput. Their results have shown that jamming at the MAC layer can be launched in an actual wireless network, becoming a potential threat to wireless networks around the world.

To stop RoQ attacks on MANETs, Arunmozhi and Venkataramani in [64] have proposed a Flow Monitoring (FMON) scheme that employs a MAC layer-based detection scheme based on the frequency and retransmission of RTS/CTS and data, as well as a response based on ECN marking. The performance of FMON has been compared to SWAN and SPA-ARA protocols. Their results have shown that FMON outperforms the other two schemes.

To detect RoQ attacks, Gulati and Dhaliwal in [119] have proposed the election of a computer to be the attack monitor. After that, the traffic is monitored and if there is a sudden increase of traffic in a short period of time above a certain threshold, all the nodes connected to that flow will be added to a list of suspects. In the Attack Monitor, the node will be added to a list called checking table if it appears countless times. This table is then sent to all nodes in the network that have verified that their traffic is above the threshold during a certain period of time. If it is positive, the node is added to the attacker table, otherwise it is removed from the checking table. All nodes in the attacker table will have their traffic blocked. Their results have shown that is possible to reduce packet loss and improve throughput.

Wen et al. in [120] have combined anomaly detection with misuse detection to detect potential anomalies through the use of wavelet multi-resolution analysis and auto-correlation analysis. The obtained experimental results show that RoQ attacks were detected accurately with both low false positive and low false negative rates.

Gang et al. in [121] have presented performance tests among MIPv4, MIPv6 and FMIPv6 against RoQ attacks with and without solution. The solution was to use a Hamilton-path-based scheme to decrease the packet loss of the traffic nodes. In the scenario without the proposed solution, the mobile protocol suffers more damage than in the scenario with the proposed solution. Their results have shown that the Hamilton-path-based scheme can decrease packet loss and delay in overlay networks.

Hongsong et al. in [122] have created a novel detection method based on the union of the HHT with EEMD and the Correlated Coefficient Method. This mix of methods aims at eliminating possible problems related to the signal generated by the attack traffic such as mode mixing and false components. To generate the RoQ attack traffic, the Ad hoc Ad hoc On-demand Multipath Distance Vector (AOMDV) was used with flood Route REQuest (RREQ) messages. Their results have shown a highly efficient detection of the attacks.

3.4 Proposed Methods for Detecting RoQ Attacks

This section identifies the set of three features to be jointly used by each classifier and describes two different approaches for detection of RoQ DDoS attacks. The first approach, described in subsection 3.4.2, consists on the separate use of eleven machine learning algorithms. The second approach consists on the use of a combination of three distinct methods: FL, MLP and ED. This second approach is described in subsection 3.4.3.

3.4.1 Classification Features

Given a set of candidate features, the problem of feature selection consists on the selection of a feature or a subset of features that performs the best under some classification algorithms. This process can reduce not only the cost of recognition by reducing the number of features, but also provide a better classification accuracy due to finite dataset size effects [212]. Although the feature selection process is useful when we have a large number of features [213, 214, 215], it also may impose severe restrictions for real-time operation. Therefore, in this chapter, we followed an heuristic approach of manually selecting features that lead to good classification performance without compromising real-time operation, as followed in other approaches, such as in [129, 216] where the authors used the number of packets for detecting DDoS attacks, [93, 188, 217, 218, 219] where the authors used entropy for DDoS attack detection, [220] where the authors used average inter-arrival time in a defence mechanism against TCP SYN flood DDoS attack, or [221] where the authors used a combination of entropy, number of packets and average inter-arrival time for detection of encrypted peer-to-peer traffic.

We consider three features for classification purposes: number of packets, entropy and average of inter-arrival time. Since, in our preliminary research study, each of these three features used individually led to poor classification results, except entropy, which for some cases led to good classification results as we can see in Section 3.6.3, we explore in this chapter the joint use of these three features for RoQ attack detection. The number of packets in a given flow is chosen as a feature because the amount of data flows may grow substantially even in a low-rate DoS attack.

The entropy measures the degree of uncertainty information associated with a random variable [222]. The more uncertain the result of a random experiment is, the more information is obtained by observing its occurrence. In this work, we evaluate the entropy of the quintet consisting of source IP address, source port, destination IP address, destination Port and transport protocol (TCP and UDP) for the network traffic. Therefore, the higher the randomness of the source IP address and the source port fields the greater the entropy and conversely, the more constant the source IP address and the source port

fields, the smaller the entropy. The entropy H is given by [222]:

$$H = -\sum_{i=1}^{n} p_i \log(p_i) \quad (3.2)$$

where p_i is the probability of each quintet element occurring in the time window under evaluation of the traffic trace and n is the total number of packets in the time window of the trace. If the flow does not have source neither destination port fields in the quintet, it is only composed of source and destination IP addresses and protocol in a given time window of the trace.

The third feature, the average of inter-arrival time of packets is selected because in flood attacks the packets do not wait in a queue to be sent by the network card. Instead, they are sent as quickly as possible, which result in packets with smaller RTT than legitimate packets.

These three features are computed as follows. For a certain traffic trace, we apply a sliding time window with length ΔT over the time in the Time Stamp field of the trace in the tshark tool. In this work, ΔT assumes a default value of 1 s. At the beginning, for a sliding time window with length of 1 s, we evaluate these three features for all packets in the first second of the traffic trace. Therefore, we count the number of packets within the first second of the trace, we evaluate the entropy as described above for the packets within the first second of the dataset, and we evaluate the average of inter-arrival times of packets within the first second in the trace. Then, we slide the time window 1 s forward, and evaluate the number of packets, entropy and average of inter-arrival times for the packets within the second second (sliding time window with length of 1 s) of the trace. This process is repeated by sliding the time window with length ΔT until it reaches the end of the dataset. The last time window may have a length smaller than ΔT because the length of the traces usually is not a multiple of ΔT.

Using the above procedure, for a given traffic trace, we obtain, for all time windows with length ΔT, the values of the three features for the packets within each time window of the traffic trace. The set of these three features for all time windows of the trace is named traffic dataset. For performance assessment purposes of the classifiers, besides the values of the three features for each time window, we also have the information about the traffic (legitimate or attack) per time window to serve as ground-truth, having this information been obtained during the construction of the traffic trace in a controlled environment. The classification approaches described along next subsections uses the traffic dataset to classify the traffic.

3.4.2 Machine Learning Algorithms and Their Settings

The following machine learning algorithms are considered in this work to investigate the detection of RoQ attacks: MLP with backpropagation, K-NN, SVM and MNB, GNB, LR, DT, ADABOOST, XGB, RF and LGBM.

Table 3.2: Machine learning algorithms and their settings.

Algorithm	Settings
K-NN	n_neighbors=5, weights=uniform, algorithm=auto, leaf_size=30, p=2, metric=minkowski, metric_params=None, n_jobs=None
MLP	activation=relu, alpha=1e-05, batch_size=auto, beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(5, 2), learning_rate=constant, learning_rate_init=0.001, max_iter=200, momentum=0.9, nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True, solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False
SVM	kernel=rbf, degree=3, gamma=auto, coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache_size=200, verbose=False, max_iter=-1
MNB	alpha=1.0, fit_prior=True, class_prior=None
GNB	priors=None, var_smoothing=1e-09
LR	C=1.0, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, l1_ratio=None, max_iter=100, multi_class='auto', n_jobs=None, penalty='l2', random_state=1, solver='lbfgs', tol=0.0001, verbose=0, warm_start=False
DT	ccp_alpha=0.0, class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, random_state=1, splitter='best'
ADABOOST	algorithm='SAMME.R', base_estimator='deprecated', estimator=None, learning_rate=1.0, n_estimators=50, random_state=1
XGB	ccp_alpha=0.0, criterion='friedman_mse', init=None, learning_rate=0.1, loss='log_loss', max_depth=3, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_iter_no_change=None, random_state=1, subsample=1.0, tol=0.0001, validation_fraction=0.1, verbose=0, warm_start=False
RF	bootstrap=True, ccp_alpha=0.0, class_weight=None, criterion='gini', max_depth=None, max_features='sqrt', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None, oob_score=False, random_state=1, verbose=0, warm_start=False
LGBM	boosting_type='gbdt', class_weight=None, colsample_bytree=1.0, importance_type='split', learning_rate=0.1, max_depth=-1, min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0, n_estimators=100, n_jobs=None, num_leaves=31, objective=None, random_state=1, reg_alpha=0.0, reg_lambda=0.0, subsample=1.0, subsample_for_bin=200000, subsample_freq=0, verbose=-1

These eleven algorithms are separately used to classify Internet traffic as legitimate traffic or attack traffic, jointly with three features evaluated for each traffic dataset. The implementation of these eleven algorithms at scikit-learn [223] has been used with their default configurations. The eleven algorithms and their settings used in this work are summarized in Table 3.2. The training dataset, with the values of the three features per time window for the training trace described in Section 5, has been used for the training phase of these algorithms.

3.4.3 Approach Based on Fuzzy Logic, MLP and Euclidean Distance

This approach is based on a combination of three methods: FL, MLP and ED. Fuzzy logic was introduced by Zadeh in 1965 [224]. It is a mathematical theory applied to vague concepts that admit intermediate logical values between false and true (0 or 1) in regard to elements belonging to a certain set with a certain pertinence degree, giving a mathematical treatment to subjective linguistic terms.

In order to explain how a fuzzy expert system works, we divide it into five main parts, as illustrated in Figure 3.2. The first part is the input, which receives the numerical data in which the system relies on to make decisions. The second part is the fuzzification, which

transforms the input data into fuzzy information. The third part is the fuzzy inference module, which includes the knowledge base and the logical decision maker. The fourth part is the defuzzification which transforms the fuzzy inference system output into numerical information presented at the output of the system.

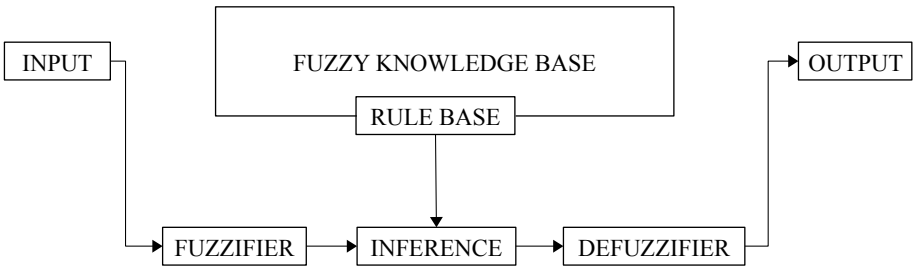

Figure 3.2: Schematic representation of fuzzy expert system (adapted from [8]).

The fuzzification is the process of normalizing the input data through pertinence functions, turning quantitative values into qualitative values such as ”very low”, ”low”, ”medium”, ”high”, ”very high” in the universe of discourse of a certain input variable. Therefore, the normalized datum (pertinence degree) can belong to more than one linguistic term, i. e., sometimes the same input datum can be classified, for example, as ”low” and ”medium” at the same time.

The inference module constructs rules that are presented in the form ”if ... then”, describing the action to be taken in response to several fuzzy normalized data outputs. It has the objective of creating a knowledge base of rules to help in the decision making, in order to obtain an accurate final result. Finally, the defuzzification process of the output data of the inference module is performed by one of the available defuzification methods to be chosen, such as first of maxima, center of area, middle of maxima and others.

In this work, the fuzzy system is configured using three linguistic terms, namely ”low”, ”medium” and ”high”, for each of the features number of packets, entropy and average inter-arrival time. The pertinence function chosen to normalize the data values is trapezoidal for all features as depicted in Figure 3.3. The range of values selected for the linguistic terms in the universe of discourse for each variable is based on the values of the features (number of packets, entropy and average inter-arrival time) evaluated from the datasets. Therefore, the lowest values of the features are for ”low” term, the highest values are for ”high” term and the average of these values are for the ”medium” term.

The trapezoidal function to normalize data is given by [8]:

$$\mu(x) = \begin{cases} \frac{x-a}{b-a},\ if\ a \leqslant x < b, \\ 1,\ if\ b \leqslant x < c, \\ \frac{d-x}{d-c},\ if\ c \leqslant x < d, \\ 0, \text{otherwise.} \end{cases} \tag{3.3}$$

where $\mu(x)$ is the normalized data, x is the data extracted from the datasets and a, b, c and

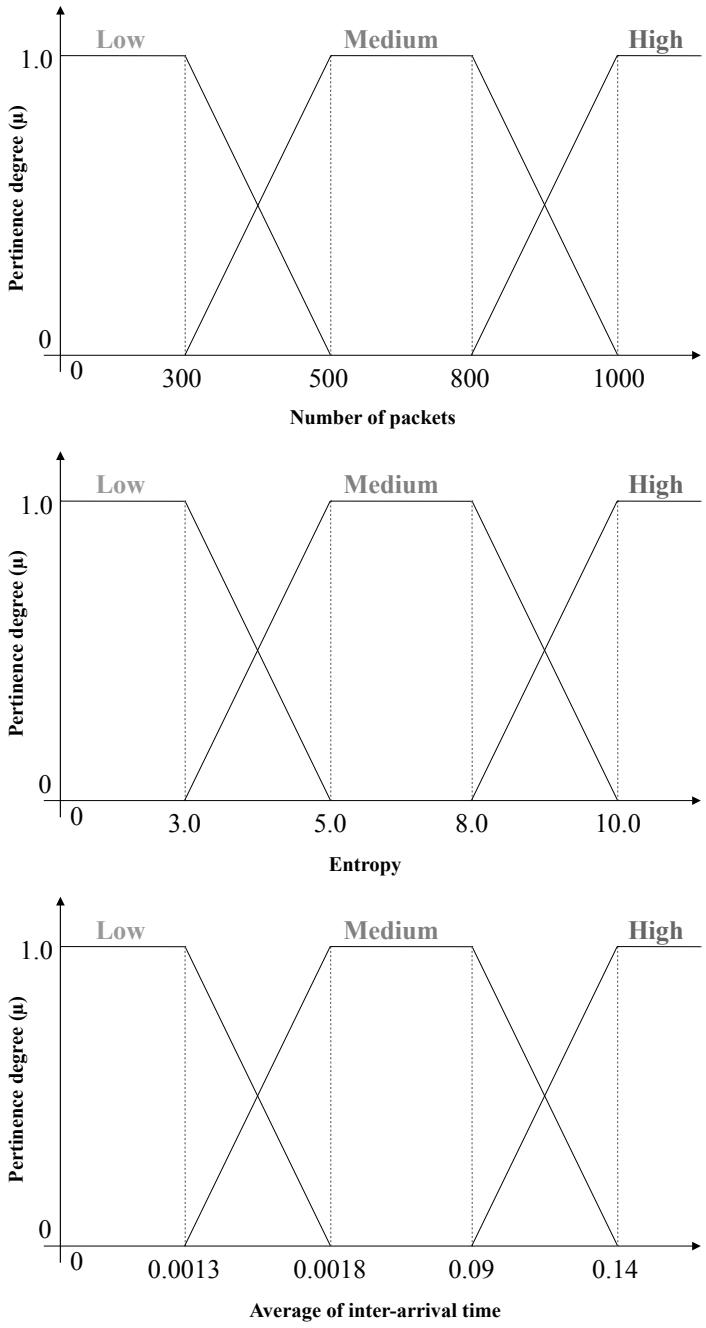

Figure 3.3: Pertinence function for the features Number of packets, Entropy and Average of inter-arrival time (ms).

d are the values referring to the data on x axis belonging to highest and lowest pertinence degree. After fuzzification, the values of the pertinence degree generated in each input variable are passed to the set of fuzzy rules. This is formed by 27 rules, which must cover all possible situations of the behaviour of the fuzzy system, as we can see in Figure 3.4. In all of the rules, the Mamdani inference model [225] is applied, in which the logical

operator “AND” is used over the antecedents of each rule, being the lowest value chosen as a consequent among the values of the pertinence degree of the triggered rule.

RULE	IF									THEN
	NUMBER OF PACKETS			ENTROPY			AVERAGE OF INTER-ARRIVAL TIME			CLASSIFICATION
	LOW	MEDIUM	HIGH	LOW	MEDIUM	HIGH	LOW	MEDIUM	HIGH	
1			X			X	X			ATTACK
2			X		X		X			ATTACK
3			X	X			X			LEGITIMATE
4			X			X		X		ATTACK
5			X		X			X		ATTACK
6			X	X				X		LEGITIMATE
7			X			X			X	ATTACK
8			X		X				X	ATTACK
9			X	X					X	LEGITIMATE
10		X				X	X			ATTACK
11		X			X		X			LEGITIMATE
12		X		X			X			LEGITIMATE
13		X				X		X		ATTACK
14		X			X			X		ATTACK
15		X		X				X		LEGITIMATE
16		X				X			X	ATTACK
17		X			X				X	LEGITIMATE
18		X		X					X	LEGITIMATE
19	X					X	X			LEGITIMATE
20	X				X		X			LEGITIMATE
21	X			X			X			LEGITIMATE
22	X					X		X		ATTACK
23	X				X			X		LEGITIMATE
24	X			X				X		LEGITIMATE
25	X					X			X	ATTACK
26	X				X				X	LEGITIMATE
27	X			X					X	LEGITIMATE

Figure 3.4: Set of base rules of the inference module.

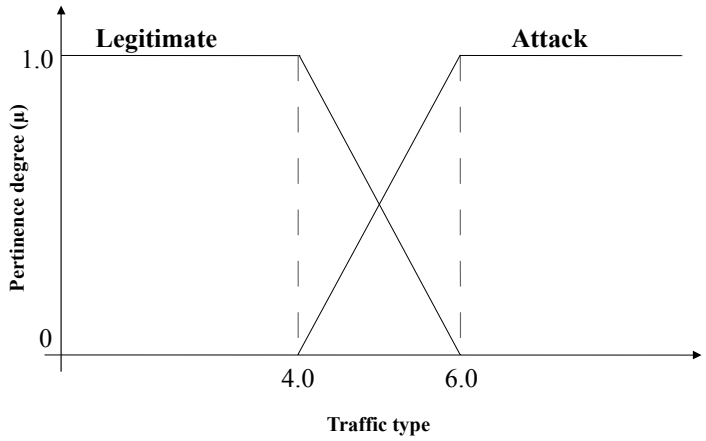

Figure 3.5: Defuzzified values of traffic type as legitimate or attack.

At the end, the defuzzification module starts after all the rules were triggered by the inference module. In order to transform the values of the pertinence degree, selected as consequent by the inference module, into an accurate output of numerical values, it is necessary to deffuzify them. For this, the Center-of-area method is used, because it is one of the most used in fuzzy expert systems. In this way, based on the generated results, the system can classify the type of traffic flow. The Center-of-area X^* is given by [8]:

$$X^* = \frac{\sum (maximum\ value \times pertinence\ degree)}{\sum pertinence\ degree} \quad (3.4)$$

This method aims to identify the maximum value to which belongs the output value of each rule of the inference module, that is, the maximum value of the central point of each linguistic variable of the defuzzification model. After performing the defuzzification, the corresponding output is classified according to the linguistic terms "legitimate" or "attack" in the universe of discourse for the traffic type (legitimate or attack). If the output is smaller or equal than 4.0, the traffic is classified as "legitimate", otherwise, if the output is larger or equal than 6.0, it is classified as "attack". Figure 3.5 illustrates this concept.

After the classification of the traffic in the trace using the FL approach, the MLP algorithm is used, as described in the previous subsection, for a new classification of the traffic in the same trace. Among the eleven machine learning algorithms, the MLP algorithm has been chosen due to its better performance results presented in Table 3.8 on subsection 3.6.3.

Both results of the traffic trace classification, for each time window with length ΔT, obtained with FL and MLP approaches are compared. If they are equal, this is the final result of the classification. If they are different, an additional step is performed, which consists in using the ED to obtain the minimum distance between each values of the number of packets and entropy, that lead to different classifications by the two approaches, and all values of the number of packets and entropy in the training dataset. Therefore, we obtain a classification using the ED for the number of packets and another classification for the entropy, since the values of the number of packets and entropy in the training dataset correspond to known traffic previously identified as legitimate or attack. The ED between the value p_k ($k = 0, ..., s$, with s being an integer smaller than m) of a feature (number of packets or entropy) that leads to different classifications by FL and MLP in the dataset under evaluation and each of the values of the feature p_i ($i = 1, ..., m$, where m is the number of values of the feature in the training dataset) is given by:

$$D(p_k, q_i) = \sqrt{(p_k - q_i)^2}. \quad (3.5)$$

Finally, the classification of the number of packets and the classification of the entropy using the ED is compared to the classifications obtained with the fuzzy logic and neural network approaches. Since we have now four classifications, if the number of classifications of attack type is greater than the classifications as legitimate type, then the final result is an attack and vice versa.

If the two classifications obtained with ED for the number of packets and the entropy are different, then, in the set of four classifications, we have two classifications as attack and two classifications as legitimate, leading to a final result that is considered as a warning. After that, the traffic packets leading to warning alerts are analysed and compared with the blocked attack traffic packets. If both sets of traffic packets have similar characteristics in terms of the high values of entropy (entropy higher than 9.0), the warning traffic will be blocked. Otherwise, traffic packets leading to warning alerts are forwarded to the

destination. The Figure 3.6 illustrates the whole classification process for detecting RoQ attacks using the approach based on FL, MLP and ED of two features (number of packets and entropy in a given time window of the trace). The additional step required when the classifications obtained with FL and MLP are different is performed by the module Traffic Classification Module detailed in Figure 3.7.

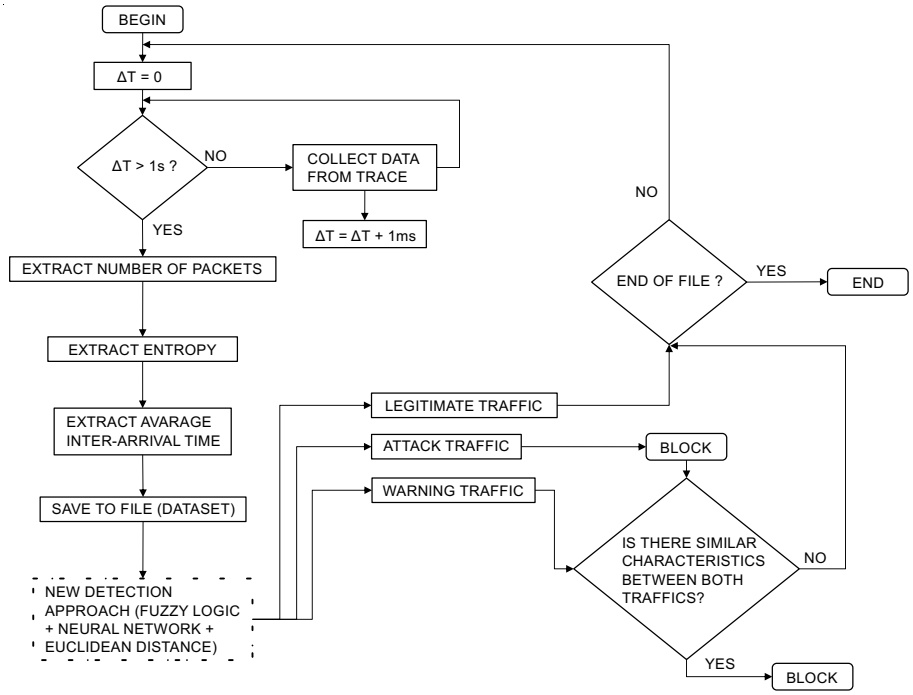

Figure 3.6: Flowchart of the classifier for detection of RoQ attacks.

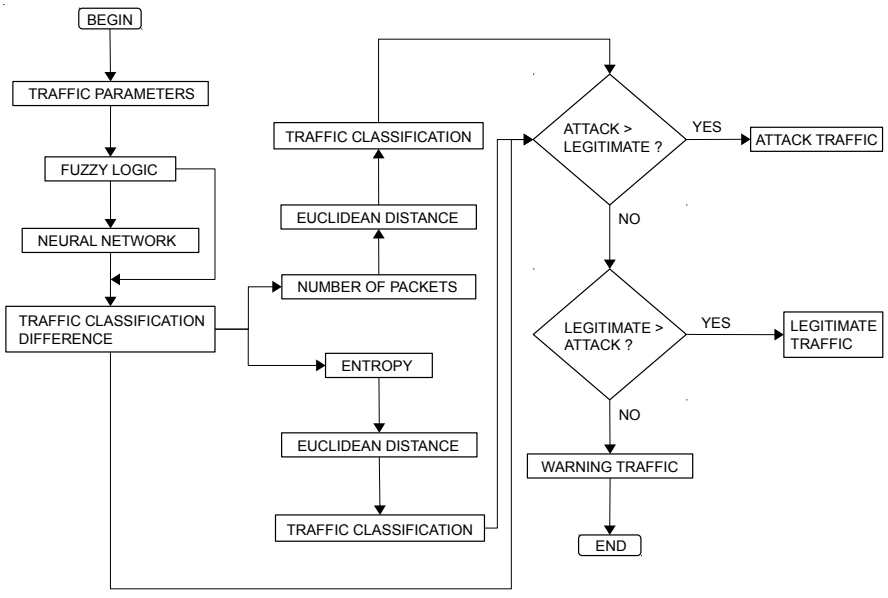

Figure 3.7: Flowchart of the classification module using FL, MLP and ED methods.

3.5 Test Environment

This section addresses traffic traces and datasets used for classification as well as the emulated and real scenarios used to obtain the traffic traces.

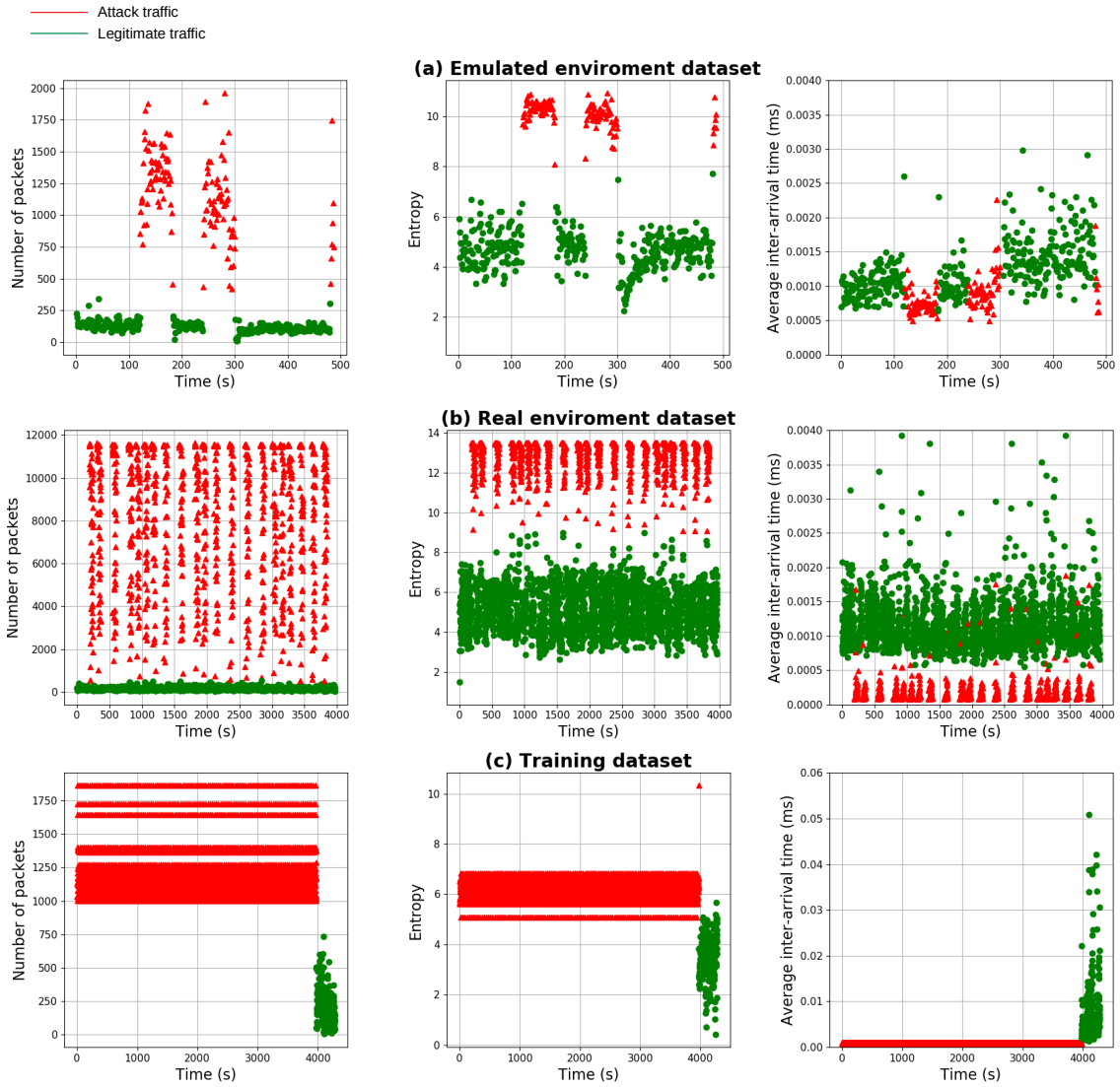

Figure 3.8: Features in the (a) emulated, (b) real and (c) training datasets for a time window ΔT=1 s.

3.5.1 Traffic Traces and Datasets

Four Internet traffic traces are considered in this work. Two traces are used for evaluation purposes and were obtained in the emulated and real environments, as described in the next subsections. The trace obtained in the emulated environment has a size of 194.8MB and the trace obtained in the real environment has a size of 11.3GB. For each of these two traces, a dataset is built with the values of the three features (number of packets, entropy and average of inter-arrival time) for each time window with length ΔT, as described in section 4.1. The values of the three features for each in these two datasets are shown

in Figure 3.8 (a) and (b). As can be seen, the number of packets and entropy are large for attack traffic and small for legitimate traffic, whereas the average inter-arrival time is smaller for attack traffic than for legitimate traffic.

The other two traces are the CAIDA [226] traffic trace, with a size of 427.6 MB used by Xiang et al. in [92] and a webserver traffic trace obtained at Instituto Federal de Educação, Ciência e Tecnologia do Tocantins (IFTO), consisting only of legitimate traffic with a size of 142.6 MB. Using firstly the CAIDA trace followed by the IFTO trace, we build only one dataset, called training dataset, which is used for the training phase of the machine learning algorithms described in previous section. Figure 3.8 (c) shows the values of the three features in the training dataset.

3.5.2 Emulated Environment

In this scenario, we used the equipment and software listed in Table 3.3 with the intent of reproducing an environment close to the real one. The software chosen to reproduce the testbed was netkit [227] since it emulates the components of real machines and the Linux operating system with many of its features. The whole process is automated and managed by shell scripts developed using the netcat tools for connecting all virtual computers. To produce the emulated trace, the attack traffic is sent by the M-RoQ attack software and the legitimate traffic is generated by curl-loader [228]. Tcpdump and tshark [229] are used to collect the traffic data.

Table 3.3. Software and hardware specification for emulated testbed.

Computer host software	
Linux	Ubuntu - version 16.04 LTS - Xenial Xerus
Netkit-ng	Core - version 3.0.4
	Filesystem - version 7.0
	Kernel - version 3.2
	Linux - version 3.2.54-netkit-ng-K3.2
Computer host hardware	
Desktop	Memory - 3GB
	Hard disk - 100GB
Netkit software	
Hping3	version 3.0.0-alpha-2
Shell bash	version 4.2.37
Netcat (nc6)	version 1.0
IPTraf	version 3.0.0
TCPDUMP	version 4.3.0
Tshark	version 1.12.1
Slowhttptest	version 1.7
Apache2	version 2.4.33
Curl-loader	version 0.56
Netkit hardware	
Legitimate clients	memory - 100MB
Attackers	memory - 128MB
Routers 1-2	memory - 100MB
Border router	memory - 256MB
Victim	memory - 512MB

The process illustrated in Figure 3.9 uses nine computers and ten routers to simulate a real Internet environment. To produce the trace, the legitimate client computers, Client 1 and

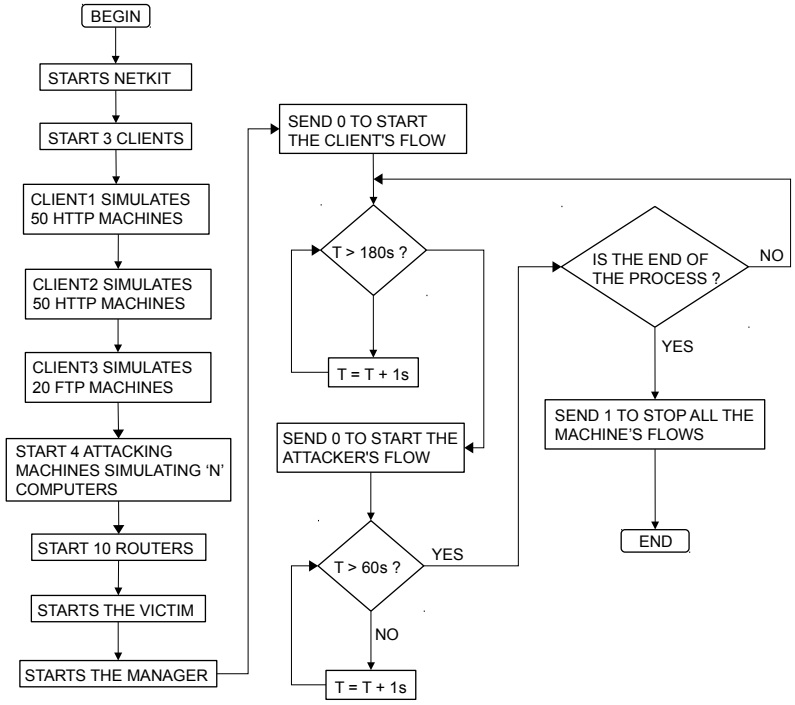

Figure 3.9. Flowchart to produce the trace in the emulated environment.

Client 2, generate HTTP traffic while Client 3 generates FTP traffic, both using curl-loader. The Manager machine starts and stops the entire process. The attacking computers use the attack period of time (ΔT) set to 1 s. The curl-loader is used to generate legitimate traffic in order to simulate the real traffic data of the Internet to test the link capacity of the server. The computers of Client 1 and Client 2 are configured to simulate 50 machines, each generating HTTP traffic. The computer of Client 3 is configured to simulate 20 machines generating FTP traffic, making a total of 120 machines transmitting traffic. The attack traffic consists of UDP packets each having a size of 1024 kbytes. The attack target computer is a Web server running Apache software.

Netkit by default starts the emulated machines with 32 MB of memory, but we set the amount of memory for each emulated machine as summarized in Table 3.3 in order to support other software that may be installed later, such as gcc, IPTraf and scripts created for this purpose. The target computer and the border router have more memory than the others because they spend more resources collecting data and processing all flows. The attacking machines have slightly more memory, because the M-RoQ software increases the memory usage for performing the RoQ attack.

3.5.3 Real Environment

This scenario is implemented at IFTO, which has a very broad network structure, using the equipment and software listed in Table 3.4. The network structure contains several routers among the classrooms, allowing data traffic from different locations thus mimic-

king the real Internet traffic.

Table 3.4. Software and hardware specification for real testbed.

Software	
Linux Ubuntu	version 16.04 LTS - Xenial Xerus
Hping3	version 3.0.0-alpha-2
Shell bash	version 4.3.48
Netcat	version 1.105-7ubuntu1
IPTraf	version 3.0.0
TCPDUMP	version 4.9.0
Tshark	version 2.2.6
Slowhttptest	version 1.7
Apache2	version 2.4.33
Curl-loader	version 0.56
Cisco 2801 IOS	version 15.0
Hardware	
Desktop	Memory - 8GB
	Hard disk - 1TB
Router Cisco 2801 series	Memory - 128MB
	FLASH memory - 32MB

This environment has the same software suite used in the emulated environment with some additions and it is composed of five computers, one server and three Cisco routers. The process illustrated in Figure 3.10 simulates a real Internet environment in a real testbed, where the legitimate Client 1 computer generates HTTP traffic while Client 2 generates FTP traffic, both using curl-loader. The attacking computers also used M-RoQ attack software with $\Delta T = 1$ s. Thus, the legitimate computer Client 1 was configured to simulate 100 machines generating HTTP traffic each and Client 2 was configured to simulate 20 machines, each one generating FTP traffic.

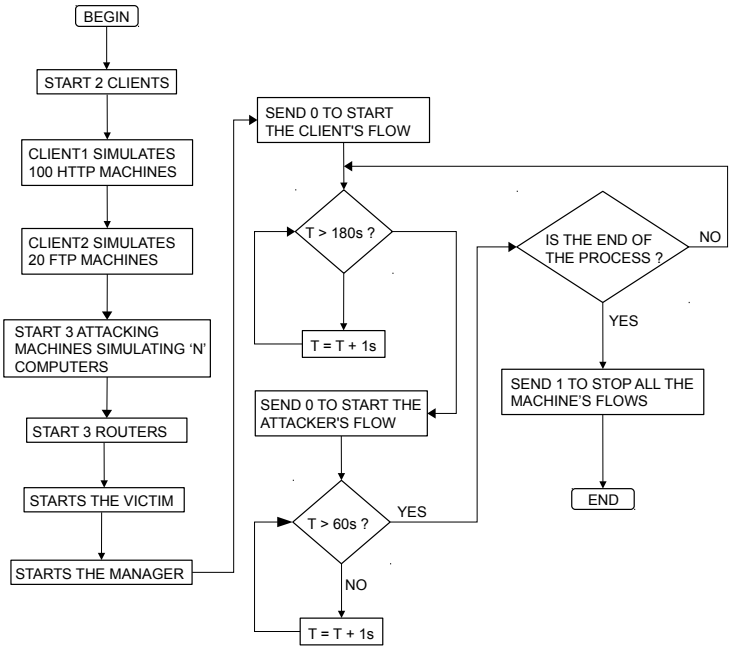

Figure 3.10. Flowchart to produce the traffic trace in the real environment.

3.5.4 M-RoQ Software

We developed Manipulated-RoQ (M-RoQ) attack software to generate RoQ-like attacks.

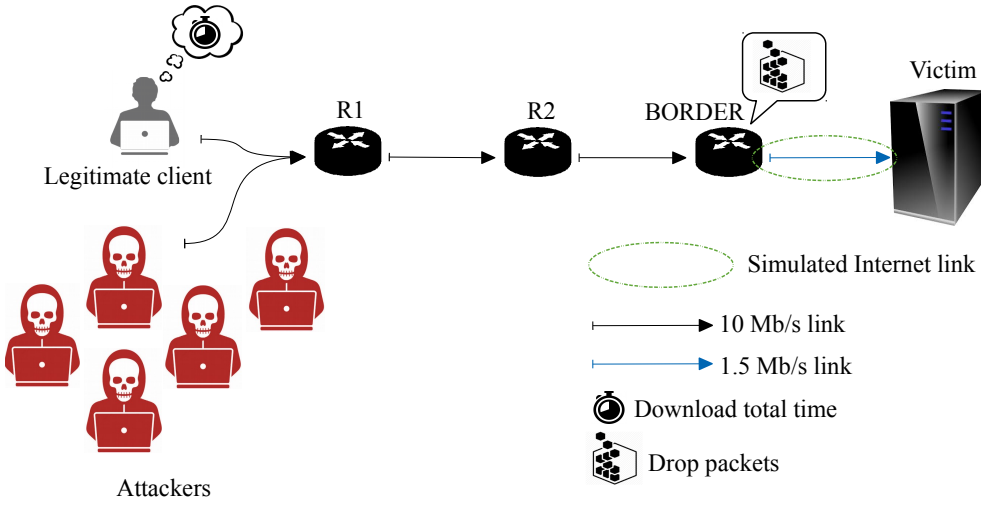

Figure 3.11. M-RoQ software testbed.

To illustrate its operation, the scenario depicted in Figure 3.11 was implemented in an emulated environment using netkit software and is composed of 10 virtual Linux computers, of which five emulate one zombie machine each, generating an attack traffic towards the target (victim) represented by the computers marked in red color.

The process begins at the end customer machine. First of all, the attacking machine, the BORDER router and the victim computer start a server socket that wait for the signal of the client running on the legitimate client machine to initiate their activities. The server socket on the attacking machines will execute the RoQ attack using the Hping3 program after receiving the signal. Since Hping3 is a DDoS attack tool, it tries to completely stop the services of the victim. Thus, we developed a software in C language that manipulates Hping3 in a RoQ attack fashion. M-RoQ software consists of the Algorithm 1, that we made available online in [230]. The M-RoQ uses the Linux timeout command to create the ON time traffic and the usleep function to generate the OFF time period (which is the time gap between the former shockwave traffic attack and the next one). The attack flow is composite of UDP (-2 option) messages with spoofed random source IP addresses mimicking the public Internet IP.

Algorithm 1 M-RoQ software

```
time ← 0
while time < 60 do
    system("timeout 0.3 hping3 --rand-source --flood -2 <dst IP>
          -p <dst PORT> -d 1024 &")
    usleep(700000)
    time++
system("killall -9 hping3")
```

The server socket in the BORDER router collects the total amount of dropped packets as well as the flows arriving on it by using the tcpdump tool. In the victim, the server socket starts the IPTraf which measures the TCP throughput. Finally, the end customer (legitimate client) client socket signals all machines and runs a shell script which collects the total download time of the FTP. All the information collected during the experiments with the attack scenario is compared to the experiments without the attack scenario to verify if the M-RoQ software is working as described and if the QoS of the victim services has been reduced.

3.6 Results and Discussion

In this section, we present the results related to all scenarios in two distinct environments as well as the M-RoQ software.

3.6.1 Performance Metrics

To evaluate the performance of the classifiers, we use precision, recall, F1-score and confusion matrix table metrics, which are defined as follow [231]:

$$Precision = \frac{TP}{TP + FP} \quad (3.6)$$

$$Recall = \frac{TP}{TP + FN} \quad (3.7)$$

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3.8)$$

where True Positive (TP) is the number of sample cases classified correctly, i.e., classifying attack traffic as an attack. The True Negative (TN) is the number of sample cases classified correctly, i.e., classifying legitimate traffic as legitimate. The False Positive (FP) is the number of positive sample cases incorrectly classified, i. e., classifying attack traffic as legitimate traffic and finally, False Negative (FN) is the number of negative sample cases incorrectly classified, i. e., classifying legitimate traffic as an attack. The F1-score metric is the harmonic mean between precision and recall. Since this measure is an average, it gives a more accurate view of the efficiency of the classifier than merely precision or recall. Therefore, these metrics are used for evaluating the performance of the proposed approaches to classify the advanced attacks considered in this work. The confusion matrix is a table that is often used to describe the performance of a classification model on a set

of test data for which the true values are known. It allows the visualization of the classifier performance.

3.6.2 Impact of RoQ Attack Period on Quality of Service

In this subsection, we investigate the impact of the attack period on the quality of service QoS of the target (victim) using the M-RoQ software.

Table 3.5. QoS parameters with and without RoQ attacks for attack periods of T=1 s, T=2 s, and T=3 s. D. Time: Download time.

T=1 s	**Mean**	**Max**	**Min**	**Standard Deviation**	**Margin of error**	**Confidence Interval**
Throughput (kb/s)	27.65	52.57	7.81	7.77	0.12	27.53 to 27.77
Delay (ms)	34.14	107.6	16.92	11.55	0.17	34.01 to 34.35
Jitter (ms)	63.08	124.91	33.84	13.66	0.16	62.91 to 63.24
Drop packets	3198.20	9676	240	1839.61	1.09	3197.11 to 3199.29
D. Time (s)	136.37	432	64	48.40	0.40	135.97 to 136.77
T=2 s	**Mean**	**Max**	**Min**	**Standard Deviation**	**Margin of error**	**Confidence Interval**
Throughput (kb/s)	79.39	110.72	54.10	9.02	0.10	79.30 to 79.49
Delay (ms)	12.16	17.39	8.75	1.34	0.02	12.14 to 12.18
Jitter (ms)	24.21	31.30	17.50	2.46	0.04	24.18 to 24.25
Drop packets	572.39	1628	30	331.79	0.97	571.42 to 573.36
D. Time (s)	42.52	60	31	4.68	0.06	42.46 to 42.58
T=3 s	**Mean**	**Max**	**Min**	**Standard Deviation**	**Margin of error**	**Confidence Interval**
Throughput (kb/s)	113.89	124.88	94.53	7.28	0.07	113.83 to 113.96
Delay (ms)	8.53	10.24	7.52	0.60	0.01	8.52 to 8.54
Jitter (ms)	17.03	20.48	15.04	1.16	0.02	17.02 to 17.05
Drop packets	396.08	1807	2	306.92	1.21	394.87 to 397.29
D. Time (s)	29.39	36	27	2.08	0.03	29.36 to 29.42
Without attack	**Mean**	**Max**	**Min**	**Standard Deviation**	**Margin of error**	**Confidence Interval**
Throughput (kb/s)	176.61	187.88	168.11	3.03	0.02	176.59 to 176.63
Delay (ms)	5.57	5.78	5.46	0.15	0	5.57 to 5.58
Jitter (ms)	11.15	11.55	10.93	0.29	0.01	11.14 to 11.15
Drop packets	0	0	0	0	0	0
D. Time (s)	18.92	20	18	0.31	0.01	18.91 to 18.93

Therefore, we define three scenarios with different attack periods T and one scenario without attack. Since a RoQ attack does not have the same attack characteristic than a Shrew attack, i. e., it does not have to try to identify the RTO of the TCP flow, we use three different attack periods with T=1 s, T=2 s, and T=3 s, in order to investigate their impact on the reduction of the QoS of the victim. For each attack scenario, we consider a set of 100 experiments which consists in running 100 times the M-RoQ software plus 100 downloads of a file. For the scenario without attack, the set of 100 experiments consists only in 100 downloads of a file. For each collected parameter, we present the mean, the maximum, the minimum, the standard deviation and the margin of error values based on

a confidence interval of 95% as summarized in Table 3.5.

The results depicted in Figure 3.12 show that the QoS of the victim is compromised for the three attack time periods. As can be seen in this figure, all the graphics with purple color have the best QoS for the applications because there is no attack traffic, while the QoS for the other colors are decreasing as the period of time T decreases. The attack with T=1 s leads to the larger reduction of the QoS because it is the shortest period of time before the next shockwave traffic flood. Thus, the longer the period of time before the next shockwave traffic flood, the faster will be the recovery of the end system protocols in adapting their sending rates.

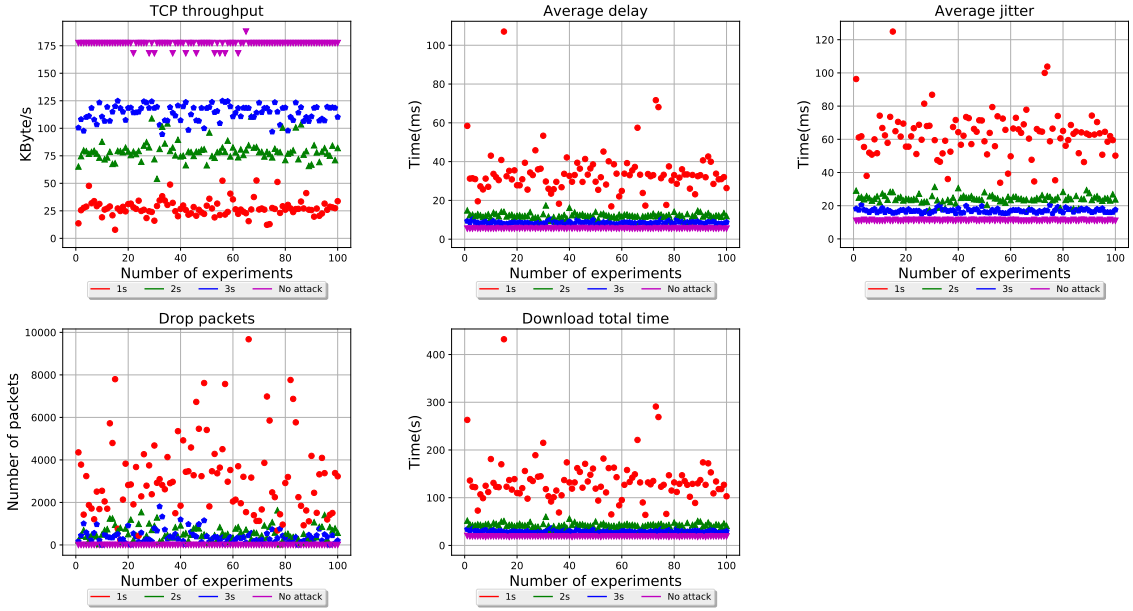

Figure 3.12. QoS parameters as a function of the number of experiments with and without RoQ attacks for attack periods of T=1 s, T=2 s, and T=3 s.

Additionally, the TCP protocol is more impacted in the experiment with T=1 s because when it tries to recover from the packet loss after the RTO, it will more quickly face another shockwave traffic flood forcing it to use a new RTO.

3.6.3 Performance of the Classification Approaches

This section provides an evaluation of the classification approaches for the traffic traces obtained in emulated and real environments. In Tables 3.6, 3.7 and 3.8, we provide the evaluation of the eleven machine learning algorithms using each feature separately: number of packets, average inter-arrival time and entropy, respectively.

In Table 3.9, we provide the evaluation of the machine learning algorithms using the three features, i.e. number of packets, entropy and average of inter-arrival times, and in Table 3.12, we provide the results of the evaluation of the proposed approach based on FL, MLP

and ED.

As can be seen in Tables 3.6, 3.7, 3.8 and 3.9, the choice of the features can deeply impact the performance of the classification. As we can see in Tables 3.6, 3.7 and 3.8, the separate use of each of the three features leads to poor classification results. This is due to the fact that the features have values that are, at some points, very close to each other, causing the algorithm to incorrectly classify the data for that feature. In this case, the classifier with the joint use of the three features leads to a better performance for detecting attacks than the classifier with one feature. This can also be seen in the confusion matrix column in Tables 3.6, 3.7, 3.8 and 3.9, where the amount of FP and FN in Tables 3.6, 3.7 and 3.8 is larger than the amount of FP and FN in Table 3.9. With only one exception, the GNB algorithm achieved 100.00% of accuracy in classifying the real environment traffic as can bee seen in Table 3.6.

As we can see in Table 3.9, the results of the classification for emulated traffic show that precision ranges from 42.35% (LGBM, ADABOOST) to 100% (LR, SVM, XGB, DT, K-NN, RF, MLP) for attack traffic and from 87.71% (LR) to 98.62% (MLP) for legitimate traffic and recall ranges from 61.53% (LR) to 96.15% (MLP) for attack traffic and from 54.62% (LR) to 100.00% (LR, SVM, XGB, DT, K-NN, RF, MLP) for legitimate traffic. The results of the classification for real traffic show that precision ranges from 41.91% (LGBM, ADABOOST) to 100% (LR, XGB, DT, SVM, K-NN, RF, MLP) for attack traffic and from 99.13% (LGBM, ADABOOST) to 99.89% (MLP) for legitimate traffic and recall ranges from 99.16% (LGBM, ADABOOST) to 99.75% (MLP) for attack traffic and from 41.16% (LGBM, ADABOOST) to 100% (LR, XGB, DT, SVM, K-NN, RF, MLP) for legitimate traffic.

We also show in Table 3.9 that F1-score ranges from 57.91% (LGBM, ADABOOTS) to 98.04% (MLP) for attack traffic and from 69.27% (SVM) to 99.30% (MLP) for legitimate traffic, while, for real traffic, F1-score ranges from 58.92% (MNB) to 99.87% (MLP) for attack traffic and from 58.17% (LGBM, ADABOOST) to 99.95% (MLP) for legitimate traffic. For the emulated traffic the accuracy ranges from 64.47% (LGBM, ADABOOST) to 98.97% (MLP), while, for real traffic the accuracy ranges from 58.54% (LGBM, ADABOOST) to 99.92% (MLP). According to Table 3.9, MLP leads to a slightly better performance than the other ten machine learning algorithms under study.

Table 3.12 shows the results of the performance evaluation for the proposed approach based on FL, MLP and ED. As we can see, it leads to a better performance than the eleven machine learning algorithms as shown in Table 3.9 due to the fact that the warning traffic classification leads to a decrease of false negative and false positive rates. This can be observed in Table 3.10 for the emulated environment and Table 3.11 for the real environment. The traffic marked as warning has the feature *Number of packets* smaller than 1000 packets, which is not considered as an attack, but it has a large value for the entropy, which can be considered as an attack. Therefore, this abnormality can degrade the performance of the classifier. This abnormality can be caused by two factors. The first factor

Table 3.6. Results of the performance evaluation of the eleven machine learning algorithms using number of packets for emulated and real traces.

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$			Accuracy
							attack	**legitimate**	
Emulated environment	GNB	attack	100.00%	96.15%	98.03%	**attack**	125	5	98.97%
		legitimate	98.61%	100.00%	99.30%	**legitimate**	0	357	
							attack	**legitimate**	
	LR	attack	100.00%	96.15%	98.03%	**attack**	125	5	98.97%
		legitimate	98.61%	100.00%	99.30%	**legitimate**	0	357	
							attack	**legitimate**	
	LGBM	attack	100.00%	93.07%	96.41%	**attack**	121	9	98.15%
		legitimate	97.54%	100.00%	98.75%	**legitimate**	0	357	
							attack	**legitimate**	
	K-NN	attack	100.00%	88.46%	93.88%	**attack**	115	15	96.91%
		legitimate	95.97%	100.00%	97.94%	**legitimate**	0	357	
							attack	**legitimate**	
	ADABOOST	attack	100.00%	84.61%	91.66%	**attack**	110	20	95.89%
		legitimate	94.69%	100.00%	97.27%	**legitimate**	0	357	
							attack	**legitimate**	
	DT	attack	100.00%	84.61%	91.66%	**attack**	110	20	95.89%
		legitimate	94.69%	100.00%	97.27%	**legitimate**	0	357	
							attack	**legitimate**	
	RF	attack	100.00%	84.61%	91.66%	**attack**	110	20	95.89%
		legitimate	94.69%	100.00%	97.27%	**legitimate**	0	357	
							attack	**legitimate**	
	XGB	attack	100.00%	84.61%	91.66%	**attack**	110	20	95.89%
		legitimate	94.69%	100.00%	97.27%	**legitimate**	0	357	
							attack	**legitimate**	
	SVM	attack	100.00%	2.31%	4.51%	**attack**	3	127	71.71%
		legitimate	73.76%	100.00%	84.90%	**legitimate**	0	357	
							attack	**legitimate**	
	MNB	attack	26.69%	100.00%	42.14%	**attack**	130	0	26.69%
		legitimate	0%	0%	0%	**legitimate**	357	0	
							attack	**legitimate**	
	MLP	attack	26.69%	100.00%	42.14%	**attack**	130	0	26.69%
		legitimate	0%	0%	0%	**legitimate**	357	0	

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$			Accuracy
							attack	**legitimate**	
Real environment	GNB	attack	100.00%	100.00%	100.00%	**attack**	1189	0	100.00%
		legitimate	100.00%	100.00%	100.00%	**legitimate**	0	2777	
							attack	**legitimate**	
	K-NN	attack	100.00%	99.16%	99.58%	**attack**	1179	10	99.96%
		legitimate	99.64%	100.00%	99.82%	**legitimate**	0	2777	
							attack	**legitimate**	
	LR	attack	99.83%	100.00%	99.91%	**attack**	1189	0	99.94%
		legitimate	100.00%	99.92%	99.96%	**legitimate**	2	2775	
							attack	**legitimate**	
	LGBM	attack	100.00%	99.49%	99.74%	**attack**	1183	6	99.84%
		legitimate	99.78%	100.00%	99.89%	**legitimate**	0	2777	
							attack	**legitimate**	
	DT	attack	100.00%	98.99%	99.49%	**attack**	1177	12	99.69%
		legitimate	99.56%	100.00%	99.78%	**legitimate**	0	2777	
							attack	**legitimate**	
	RF	attack	100.00%	98.99%	99.49%	**attack**	1177	12	99.69%
		legitimate	99.56%	100.00%	99.78%	**legitimate**	0	2777	
							attack	**legitimate**	
	XGB	attack	100.00%	98.99%	99.49%	**attack**	1177	12	99.69%
		legitimate	99.56%	100.00%	99.78%	**legitimate**	0	2777	
				100.00%			**attack**	**legitimate**	
	ADABOOST	attack	100.00%	98.99%	99.49%	**attack**	1177	12	99.69%
		legitimate	99.56%	100.00%	99.78%	**legitimate**	0	2777	
							attack	**legitimate**	
	SVM	attack	100.00%	0.17%	0.34%	**attack**	2	1187	95.90%
		legitimate	70.06%	100.00%	82.39%	**legitimate**	0	2777	
							attack	**legitimate**	
	MNB	attack	29.98%		46.13%	**attack**	1189	0	29.97%
		legitimate	0%	0%	0%	**legitimate**	2777	0	
							attack	**legitimate**	
	MLP	attack	29.98%	100.00%	46.13%	**attack**	1189	0	29.97%
		legitimate	0%	0%	0%	**legitimate**	2777	0	

Table 3.7. Results of the performance evaluation of the eleven machine learning algorithms using average inter-arrival time for emulated and real traces.

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$			Accuracy
Emulated environment	K-NN	attack	100.00%	88.46%	93.88%		**attack**	**legitimate**	96.91%
		legitimate	95.97%	100.00%	97.94%	**attack**	115	15	
						legitimate	0	357	
	LGBM	attack	42.34%	91.53%	57.90%		**attack**	**legitimate**	64.47%
		legitimate	94.66%	54.62%	69.27%	**attack**	119	11	
						legitimate	162	195	
	ADABOOST	attack	42.34%	91.53%	57.90%		**attack**	**legitimate**	64.47%
		legitimate	94.66%	54.62%	69.27%	**attack**	119	11	
						legitimate	162	195	
	DT	attack	42.34%	91.53%	57.90%		**attack**	**legitimate**	64.47%
		legitimate	94.66%	54.62%	69.27%	**attack**	119	11	
						legitimate	162	195	
	XGB	attack	42.34%	91.53%	57.90%		**attack**	**legitimate**	64.47%
		legitimate	94.66%	54.62%	69.27%	**attack**	119	11	
						legitimate	162	195	
	RF	attack	42.34%	91.53%	57.90%		**attack**	**legitimate**	64.47%
		legitimate	94.66%	54.62%	69.27%	**attack**	119	11	
						legitimate	162	195	
	GNB	attack	35.79%	96.92%	52.28%		**attack**	**legitimate**	52.77%
		legitimate	97.03%	36.69%	53.25%	**attack**	126	4	
						legitimate	226	131	
	SVM	attack	26.69%	100.00%	42.14%		**attack**	**legitimate**	26.69%
		legitimate	0%	0%	0%	**attack**	130	0	
						legitimate	357	0	
	LR	attack	26.69%	100.00%	42.14%		**attack**	**legitimate**	26.69%
		legitimate	0%	0%	0%	**attack**	130	0	
						legitimate	357	0	
	MNB	attack	26.69%	100.00%	42.14%		**attack**	**legitimate**	26.69%
		legitimate	0%	0%	0%	**attack**	130	0	
						legitimate	357	0	
	MLP	attack	26.69%	100.00%	42.14%		**attack**	**legitimate**	26.69%
		legitimate	0%	0%	0%	**attack**	130	0	
						legitimate	357	0	

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$			Accuracy
Real environment	K-NN	attack	100.00%	99.16%	99.58%		**attack**	**legitimate**	99.96%
		legitimate	99.64%	100.00%	99.82%	**attack**	1179	10	
						legitimate	0	2777	
	LGBM	attack	41.91%	99.15%	58.92%		**attack**	**legitimate**	58.54%
		legitimate	99.13%	41.15%	58.16%	**attack**	1179	10	
						legitimate	1634	1143	
	DT	attack	41.91%	99.15%	58.92%		**attack**	**legitimate**	58.54%
		legitimate	99.13%	41.15%	58.16%	**attack**	1179	10	
						legitimate	1634	1143	
	XGB	attack	41.91%	99.15%	58.92%		**attack**	**legitimate**	58.54%
		legitimate	99.13%	41.15%	58.16%	**attack**	1179	10	
						legitimate	1634	1143	
	ADABOOST	attack	41.91%	99.15%	58.92%		**attack**	**legitimate**	58.54%
		legitimate	99.13%	41.15%	58.16%	**attack**	1179	10	
						legitimate	1634	1143	
	RF	attack	41.91%	99.15%	58.92%		**attack**	**legitimate**	58.54%
		legitimate	99.13%	41.15%	58.16%	**attack**	1179	10	
						legitimate	1634	1143	
	GNB	attack	1.68%	2.94%	2.14%		**attack**	**legitimate**	19.31%
		legitimate	38.77%	26.32%	31.35%	**attack**	35	1154	
						legitimate	2046	731	
	LR	attack	29.98%	100.00%	46.13%		**attack**	**legitimate**	29.97%
		legitimate	0%	0%	0%	**attack**	1189	0	
						legitimate	2777	0	
	SVM	attack	29.98%	100.00%	46.13%		**attack**	**legitimate**	29.97%
		legitimate	0%	0%	0%	**attack**	1189	0	
						legitimate	2777	0	
	MNB	attack	29.98%	100.00%	46.13%		**attack**	**legitimate**	29.97%
		legitimate	0%	0%	0%	**attack**	1189	0	
						legitimate	2777	0	
	MLP	attack	29.98%	100.00%	46.13%		**attack**	**legitimate**	29.97%
		legitimate	0%	0%	0%	**attack**	1189	0	
						legitimate	2777	0	

Table 3.8. Results of the performance evaluation of the eleven machine learning algorithms using entropy for emulated and real traces.

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$			Accuracy
							attack	**legitimate**	
Emulated environment	K-NN	attack	100.00%	88.46%	93.88%	**attack**	115	15	96.91%
		legitimate	95.97%	100.00%	97.94%	**legitimate**	0	357	
							attack	**legitimate**	
	LGBM	attack	68.78%	100.00%	81.50%	**attack**	130	0	87.88%
		legitimate	100.00%	83.47%	90.99%	**legitimate**	59	298	
							attack	**legitimate**	
	ADABOOST	attack	68.78%	100.00%	81.50%	**attack**	130	0	87.88%
		legitimate	100.00%	83.47%	90.99%	**legitimate**	59	298	
							attack	**legitimate**	
	DT	attack	68.78%	100.00%	81.50%	**attack**	130	0	87.88%
		legitimate	100.00%	83.47%	90.99%	**legitimate**	59	298	
							attack	**legitimate**	
	XGB	attack	68.78%	100.00%	81.50%	**attack**	130	0	87.88%
		legitimate	100.00%	83.47%	90.99%	**legitimate**	59	298	
							attack	**legitimate**	
	RF	attack	58.82%	100.00%	74.07%	**attack**	130	0	81.31%
		legitimate	100.00%	74.50%	85.39%	**legitimate**	91	266	
							attack	**legitimate**	
	SVM	attack	48.49%	99.23%	65.15%	**attack**	129	1	71.66%
		legitimate	99.54%	61.62%	76.12%	**legitimate**	137	220	
							attack	**legitimate**	
	GNB	attack	5.26%	3.84%	4.44%	**attack**	5	125	55.85%
		legitimate	68.11%	74.78%	71.29%	**legitimate**	90	267	
							attack	**legitimate**	
	LR	attack	36.61%	100.00%	53.60%	**attack**	130	0	53.79%
		legitimate	100.00%	36.97%	53.98%	**legitimate**	225	132	
							attack	**legitimate**	
	MNB	attack	26.69%	100.00%	42.14%	**attack**	130	0	26.69%
		legitimate	0%	0%	0%	**legitimate**	357	0	
							attack	**legitimate**	
	MLP	attack	26.69%	100.00%	42.14%	**attack**	130	0	26.69%
		legitimate	0%	0%	0%	**legitimate**	357	0	

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$			Accuracy
							attack	**legitimate**	
Real environment	K-NN	attack	100.00%	99.16%	99.58%	**attack**	1179	10	99.96%
		legitimate	99.64%	100.00%	99.82%	**legitimate**	0	2777	
							attack	**legitimate**	
	LR	attack	37.49%	100.00%	54.54%	**attack**	1189	0	99.94%
		legitimate	100.00%	28.62%	44.51%	**legitimate**	1982	795	
							attack	**legitimate**	
	LGBM	attack	53.43%	100.00%	69.65%	**attack**	1189	0	73.87%
		legitimate	100.00%	62.69%	77.96%	**legitimate**	1036	1741	
							attack	**legitimate**	
	DT	attack	53.43%	100.00%	69.65%	**attack**	1189	0	73.87%
		legitimate	100.00%	62.69%	77.96%	**legitimate**	1036	1741	
							attack	**legitimate**	
	XGB	attack	53.43%		69.65%	**attack**	1189	0	73.87%
		legitimate	100.00%	62.69%	77.96%	**legitimate**	1036	1741	
				100.00%			**attack**	**legitimate**	
	ADABOOST	attack	53.43%	100.00%	69.65%	**attack**	1189	0	73.87%
		legitimate	100.00%	62.69%	77.96%	**legitimate**	1036	1741	
							attack	**legitimate**	
	RF	attack	48.45%	100.00%	65.27%	**attack**	1189	0	68.10%
		legitimate	100.00%	54.44%	70.50%	**legitimate**	1265	1512	
				100.00%			**attack**	**legitimate**	
	GNB	attack	0.22%	0.25%	0.24%	**attack**	3	1186	37.24%
		legitimate	55.41%	53.07%	54.22%	**legitimate**	1303	1474	
							attack	**legitimate**	
	SVM	attack	8.87%	12.19%	10.27%	**attack**	145	1044	36.15%
		legitimate	55.25%	46.41%	50.45%	**legitimate**	1488	1289	
							attack	**legitimate**	
	MNB	attack	29.98%		46.13%	**attack**	1189	0	29.97%
		legitimate	0%	0%	0%	**legitimate**	2777	0	
							attack	**legitimate**	
	MLP	attack	29.98%	100.00%	46.13%	**attack**	1189	0	29.97%
		legitimate	0%	0%	0%	**legitimate**	2777	0	

Table 3.9. Results of the performance evaluation of the eleven machine learning algorithms using three features for emulated and real traces.

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$			Accuracy
							attack	**legitimate**	
Emulated environment	MLP	attack	100.00%	96.15%	98.04%	**attack**	125	5	98.97%
		legitimate	98.62%	100.00%	99.30%	**legitimate**	0	357	
							attack	**legitimate**	
	GNB	attack	97.96%	96.15%	96.90%	**attack**	125	5	98.35%
		legitimate	98.61%	99.16%	98.88%	**legitimate**	3	354	
							attack	**legitimate**	
	RF	attack	100.00%	90.77%	95.16%	**attack**	118	12	97.53%
		legitimate	96.75%	100.00%	98.35%	**legitimate**	0	357	
							attack	**legitimate**	
	K-NN	attack	100.00%	88.46%	93.88%	**attack**	115	15	96.91%
		legitimate	95.97%	100.00%	97.94%	**legitimate**	0	357	
							attack	**legitimate**	
	MNB	attack	94.53%	93.08%	93.80%	**attack**	121	9	96.71%
		legitimate	97.49%	98.04%	97.77%	**legitimate**	7	350	
							attack	**legitimate**	
	DT	attack	100.00%	84.62%	91.67%	**attack**	110	20	95.89%
		legitimate	94.69%	100.00%	97.28%	**legitimate**	0	357	
							attack	**legitimate**	
	XGB	attack	100.00%	84.62%	91.67%	**attack**	110	20	95.89%
		legitimate	94.69%	100.00%	97.28%	**legitimate**	0	357	
							attack	**legitimate**	
	SVM	attack	100.00%	84.62%	91.67%	**attack**	110	20	95.89%
		legitimate	94.69%	100.00%	97.28%	**legitimate**	0	357	
							attack	**legitimate**	
	LR	attack	100.00%	61.53%	76.19%	**attack**	80	50	89.73%
		legitimate	87.71%	100.00%	93.45%	**legitimate**	0	357	
							attack	**legitimate**	
	ADABOOST	attack	42.35%	91.54%	57.91%	**attack**	119	11	64.47%
		legitimate	94.66%	54.62%	69.27%	**legitimate**	162	195	
							attack	**legitimate**	
	LGBM	attack	42.35%	91.54%	57.91%	**attack**	119	11	64.47%
		legitimate	94.66%	54.62%	69.27%	**legitimate**	162	195	

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$			Accuracy
							attack	**legitimate**	
Real environment	MLP	attack	100.00%	99.75%	99.87%	**attack**	1186	3	99.92%
		legitimate	99.89%	100.00%	99.95%	**legitimate**	0	2777	
							attack	**legitimate**	
	RF	attack	100.00%	99.24%	99.62%	**attack**	1180	9	99.77%
		legitimate	99.68%	100.00%	99.84%	**legitimate**	0	2777	
							attack	**legitimate**	
	K-NN	attack	100.00%	99.16%	99.58%	**attack**	1179	10	99.74%
		legitimate	99.64%	100.00%	99.82%	**legitimate**	0	2777	
							attack	**legitimate**	
	SVM	attack	100.00%	98.99%	99.49%	**attack**	1177	12	99.69%
		legitimate	99.57%	100.00%	99.78%	**legitimate**	0	2777	
							attack	**legitimate**	
	DT	attack	100.00%	98.99%	99.49%	**attack**	1177	12	99.69%
		legitimate	99.57%	100.00%	99.78%	**legitimate**	0	2777	
							attack	**legitimate**	
	XGB	attack	100.00%	98.99%	99.49%	**attack**	1177	12	99.69%
		legitimate	99.57%	100.00%	99.78%	**legitimate**	0	2777	
							attack	**legitimate**	
	LR	attack	100.00%	97.80%	96.71%	**attack**	1179	10	99.62%
		legitimate	99.46%	100.00%	99.73%	**legitimate**	0	2777	
							attack	**legitimate**	
	MNB	attack	92.85%	99.41%	96.02%	**attack**	1182	7	97.52%
		legitimate	99.74%	96.72%	98.21%	**legitimate**	91	2686	
							attack	**legitimate**	
	GNB	attack	92.36%	99.66%	95.87%	**attack**	1185	4	97.42%
		legitimate	99.85%	96.47%	98.13%	**legitimate**	98	2679	
							attack	**legitimate**	
	ADABOOST	attack	41.91%	99.16%	58.92%	**attack**	1179	10	58.54%
		legitimate	99.13%	41.16%	58.17%	**legitimate**	1634	1143	
							attack	**legitimate**	
	LGBM	attack	41.91%	99.16%	58.92%	**attack**	1179	10	58.54%
		legitimate	99.13%	41.16%	58.17%	**legitimate**	1634	1143	

Table 3.10. Warning alerts in the emulated environment.

	Attributes/ Values	Euclidean distance classification	Fuzzy logic classification	MLP classification
Emulated environment	Number of packets - 854 Entropy - 9.68	normal attack	normal	attack
	Number of packets - 772 Entropy - 9.59	normal attack	normal	attack
	Number of packets - 851 Entropy - 9.65	normal attack	normal	attack
	Number of packets - 760 Entropy - 9.55	normal attack	normal	attack
	Number of packets - 662 Entropy - 9.36	normal attack	normal	attack
	Number of packets - 751 Entropy - 9.55	normal attack	normal	attack
	Number of packets - 422 Entropy - 8.72	attack normal	normal	attack
	Number of packets - 857 Entropy - 9.74	normal attack	normal	attack
	Number of packets - 741 Entropy - 9.53	normal attack	normal	attack
	Number of packets - 464 Entropy - 8.84	attack normal	normal	attack
	Number of packets - 662 Entropy - 9.35	normal attack	attack	normal
	Number of packets - 772 Entropy - 9.58	normal attack	normal	attack
	Number of packets - 749 Entropy - 9.54	normal attack	normal	attack

Table 3.11. Warning alerts in the real environment.

	Attributes/ Values	Euclidean distance dist. classification	Fuzzy logic classification	MLP classification
Real environment	Number of packets - 461 entropy - 8.84	attack normal	normal	attack
	Number of packets - 390 entropy - 8.60	attack normal	normal	attack
	Number of packets - 348 entropy - 8.43	attack normal	normal	attack
	Number of packets - 733 Entropy - 9.51	normal attack	normal	attack
	Number of packets - 453 entropy - 8.82	attack normal	normal	attack
	Number of packets - 321 entropy - 8.32	attack normal	normal	attack
	Number of packets - 482 entropy - 8.91	attack normal	normal	attack
	Number of packets - 350 entropy - 8.45	attack normal	normal	attack
	Number of packets - 437 entropy - 8.76	attack normal	normal	attack
	Number of packets - 502 entropy - 8.97	attack normal	normal	attack
	Number of packets - 865 Entropy - 9.75	normal attack	normal	attack
	Number of packets - 325 entropy - 8.34	attack normal	normal	attack
	Number of packets - 759 Entropy - 9.48	normal attack	normal	attack
	Number of packets - 668 Entropy - 9.38	normal attack	normal	attack
	Number of packets - 678 Entropy - 9.40	normal attack	normal	attack
	Number of packets - 322 entropy - 8.33	attack normal	normal	attack
	Number of packets - 399 entropy - 8.64	attack normal	normal	attack
	Number of packets - 799 Entropy - 9.64	normal attack	normal	attack
	Number of packets - 386 entropy - 8.59	attack normal	normal	attack
	Number of packets - 336 entropy - 8.39	attack normal	normal	attack

Table 3.12. Results of the performance evaluation of the proposed approach based on FL, MLP and ED for emulated and real traces.

	Algorithm	Traffic	Precision	Recall	F1-score	Confusion matrix $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$			Accuracy
Emulated	FL, MLP and ED	attack	100.00%	97.70%	98.80%		**attack**	**legitimate**	99.36%
		legitimate	99.20%	100.00%	99.60%	**attack**	114	3	
						legitimate	0	357	
Real	FL, MLP and ED	attack	100.00%	100.00%	100.00%		**attack**	**legitimate**	100.00%
		legitimate	100.00%	100.00%	100.00%	**attack**	1187	0	
						legitimate	0	2762	

occurs when we apply the sliding time window with length of 1 second over the traffic trace and the last packets of an attack are included in the beginning of a time window, being composed mostly by legitimate traffic, which may lead to an increase of the entropy in that time window due to the presence of some packets of the attack. The second factor can occur when an attack is in its final phase, where the attack power is weaker and the last malicious packets may be included in a time window composed mostly by legitimate traffic, which may lead to an increase of the entropy in that time window.

As we can see in Table 3.12, when using the approach based on FL, MLP and ED, for classification of emulated traffic, we obtained a precision of 100% for attack traffic and 99.20% for legitimate traffic, a recall of 97.70% for attack traffic and 100% for legitimate traffic, and a F1-score of 98.80% for attack traffic and 99.60% for legitimate traffic, leading to an accuracy of 99.36%, while, for real traffic, we obtained a a precision of 100% for attack traffic and 100% for legitimate traffic, a recall of 100% for attack traffic and 100% for legitimate traffic and a F1-score of 100% for attack traffic and 100% for legitimate traffic. For the emulated traffic the accuracy is 99.36%, while, for real traffic the accuracy is 100%, leading to an accuracy of 100%.

3.6.4 Resource Usage

The computational resources used by the eleven machine learning algorithms and the approach based on FL, MLP and ED were collected by a computer running the GNU/Linux operating system Ubuntu version 19.04 (Disco Dingo), with 8 GB of Random Access Memory (RAM), a 64-bit AMD Phenom II X4 925 processor with 4 cores and 1 thread per core.

Table 3.13 shows the resource usage by each classification approach. Time, Central Processing Unit (CPU) and RAM consumption and execution times were collected through the GNU/Linux command /usr/bin/time with -f option and with sub-option in the '%E %M %P' format. Since CPU consumption is larger than 100% it means that at least 2 cores were used to classify the whole dataset. As can be seen in this table, the performance of the proposed approach based on FL, MLP and ED was optimized and now has a much better performance, requiring 1.20 seconds and 1.35 seconds to finish the classification

for the emulated and real traffic datasets, respectively.

Table 3.13. Computational resource usage (sorted by RAM). Optimized algorithm*.

Algorithm	Emulated dataset	Real dataset
GNB	CPU: 99.10% RAM: 93.87MB Execution time: 1.03 seconds	CPU: 98.80% RAM: 94.36MB Execution time: 1.04 seconds
MNB	CPU: 98.80% RAM: 94.31MB Execution time: 1.05 seconds	CPU: 98.90% RAM: 94.82MB Execution time: 1.05 seconds
FL, MLP and ED*	CPU: 98.80% RAM: 95.26MB Execution time: 1.20 seconds	CPU: 99.20% RAM: 96.28MB Execution time: 1.35 seconds
K-NN	CPU: 99.00% RAM: 95.95MB Execution time: 1.06 seconds	CPU: 99.10% RAM: 96.31MB Execution time: 1.22 seconds
MLP	CPU: 99.10% RAM: 95.99MB Execution time: 1.15 seconds	CPU: 99.00% RAM: 96.00MB Execution time: 1.16 seconds
DT	CPU: 98.80% RAM: 95.45MB Execution time: 1.05 seconds	CPU: 98.90% RAM: 96.26MB Execution time: 1.06 seconds
XGB	CPU: 98.80% RAM: 96.75MB Execution time: 1.17 seconds	CPU: 98.70% RAM: 97.85MB Execution time: 1.17 seconds
RF	CPU: 98.10% RAM: 96.82MB Execution time: 1.08 seconds	CPU: 98.90% RAM: 97.78MB Execution time: 1.09 seconds
ADABOOST	CPU: 99.40% RAM: 96.94MB Execution time: 1.05 seconds	CPU: 98.90% RAM: 97.75MB Execution time: 1.06 seconds
LR	CPU: 98.80% RAM: 99.30MB Execution time: 1.08 seconds	CPU: 98.80% RAM: 99.50MB Execution time: 1.09 seconds
LGBM	CPU: 105.50% RAM: 100.99MB Execution time: 1.14 seconds	CPU: 107.30% RAM: 100.76MB Execution time: 1.12 seconds

3.7 Conclusion

In this chapter, we evaluated and compared eleven machine learning algorithms for the detection of RoQ attacks: LR, RF, DT, ADABOOST, LGBM, XGB, GNB, MLP, K-NN, SVM and MNB. We also proposed an approach based on a combination of three methods, FL, MLP and ED, for the detection of RoQ attacks. We evaluated these approaches using emulated and real traffic traces. To build the traffic traces, we created an emulated environment and a real environment and developed an attack tool to generate the attacks, called M-RoQ. We showed that the use of three features, namely number of packets, entropy and average inter-arrival time, leads to a better classification of the eleven machine learning algorithms than using only the entropy as a feature. We showed that, among the

eleven machine learning algorithms, MLP leads to the best classification results on the detection of RoQ attacks and that the approach based on FL, MLP and ED outperforms MLP. For future work, we plan to explore other machine learning algorithms and compare the results with other approaches.

Chapter 4

Detection of Slowloris Attacks Using Artificial Intelligence Methods[3]

The Slowloris attack, a variant of the slow DoS attack, is a stealthy threat that aims to take down web services provided by companies and institutions. It is able to pass through the traditional defense systems, due to the low amount and high latency of its attack traffic, often mimicking legitimate user traffic. Therefore, it is necessary to investigate techniques that can detect and mitigate this type of attack and simultaneously prevent legitimate user traffic from being blocked. In this work, we investigate nine machine learning algorithms for detecting Slowloris attacks, as well as a new combination based on FL, RF, and ED that we call FRE. We first generate Slowloris attack traffic traces in various environments. We then assess these algorithms under two scenarios: hyperparameters with default values and optimized hyperparameters. We show that most of these machine learning algorithms perform very well, with the random forest leading to the best classification results with test accuracy values reaching 99.52%. We also show that our FRE method outperforms all these algorithms, with test accuracy values reaching 99.8%.

4.1 Introduction

The *Slowloris attack* [37, 130] consists in sending partial HTTP requests with an encoding `\r\n` tag at the end of the header for opening connections between a computer and a targeted web server and on maintaining those connections open in order to stop or severely slow down the target. Since the appearance of this kind of attack, several solutions have been proposed for mitigating its impact. These solutions employ strategies such as the limitation of the number of connections per user or the setup of timeouts for each connection. However, these kind of limitations can impede the connections of legitimate users which go above a pre-established limit, e.g., users who access pages with numerous objects in a non-persistent HTTP connection mode [130]. Consequently, this has motivated the development of several approaches for detecting these attacks [31, 124, 125, 126, 127, 128, 129, 130, 232], rather than equally limiting all traffic. The proposed solutions nonetheless involve a high degree of complexity and require time-

[3]The content of this chapter consists of the paper published in the following venue [32] with minor modifications: Vinícius de Miranda Rios, Damien Magoni, Pedro R. M. Inácio, Mário M. Freire, Detection of Slowloris Attacks using Machine Learning Algorithms. In Proceedings of ACM SAC Conference. ACM, Avila, Spain, 10 pages. https://doi.org/10.1145/3605098.3635919

consuming responses, limited or unreported classifier performance, or were designed for specific network types (e.g., wireless mesh networks).

This chapter focuses on detecting Slowloris attacks by employing and analyzing two different approaches. The first approach involves the evaluation of nine Machine Learning (ML) algorithms for Slowloris attacks detection, specifically K-NN, GNB, MLP, SVM, DT, MNB, RF, XGB and LGBM. These nine algorithms have been widely investigated for detecting high-rate DDoS attacks, with convincing results (e.g., MLP and k-NN [194], SVM [195], MNB [199], GNB [194], DT [233], RF [234], XGB [235] and LGBM [236]). The second approach, is a novel method which combines three different algorithms: (i) Fuzzy Logic (FL), (ii) Random Forest (RF) and (iii) Euclidean Distance (ED). We have called this method FRE and it leverages the best characteristics of each algorithm. The contributions of this work are therefore threefold:

- We generate and provide four datasets of network data traffic containing distributed Slowloris attacks, collected in various real-life and emulated environments (LAN and MAN).
- We train, test and evaluate nine ML algorithms with both default and optimized hyperparameters for detecting Slowloris attacks in these datasets.
- We propose and evaluate a novel method, called FRE, built by combining three algorithms (FL, RF, and ED).

The first contribution had to be fulfilled due to the lack of datasets containing distributed Slowloris attacks among data traffic. The best known dataset containing Slowloris traffic is the CIC-IDS 2017 [237], which contains a Slowloris attack trace coming from only one source. In order to build our datasets, we have launched several simultaneous Slowloris attacks from various multiple sources.

The chapter is structured as follows: Section 4.2 presents related prior works concerning this topic. Section 4.3 details the algorithms used in our work to detect Slowloris attacks. Section 4.4 describes the design of the dataset collection process, while Section 4.5 shows the results of our evaluation experiments.

4.2 Related Work

Although machine learning methods have been extensively investigated as a means of detecting traditional high-rate DoS/DDoS attacks, few papers have focused on DoS attacks that purposely use low-rate traffic attacks such as Slowloris attacks. Table 4.1 compares the main features and characteristics of the related works briefly discussed below.

In [124], Aiello et al. investigated the detection of anomalous traffic patterns and proposed a SBID approach to detect Slowloris attacks. The study employs statistical ap-

proaches based on signatures for IP traffic classification, leveraging machine learning techniques for flow clustering and traffic classification. One of the main threats highlighted is the Slowloris attack. The detection methodology involves comparing unknown and potentially anomalous traffic to a representation of legitimate traffic. Their research showed that the use of an average-based method, although time-consuming, provides near-absolute accuracy in detecting anomalous traffic. Dantas et al. [125] proposed a defense mechanism called SeVen, which uses the notion of state in the messages of the HTTP protocol to detect Slowloris attacks. In [126], Mongelli et al. proposed a detection method that uses the Fast Fourier Transform in the current time horizon in the frequency domain to identify the attack. Katkar et al. [127] configured an IDS system with a Naive Bayesian classifier with different methods for data pre-processing for detecting DoS/DDoS attacks against HTTP servers. Aqil et al. [128] proposed a detection algorithm that checks various features against high and low thresholds. If certain features exceed their high thresholds while others remain below their low thresholds, an attack is flagged. In [31], Hirakawa et al. described a 3-step detection scheme. The first one checks if the number of connections that are monitored is greater than a threshold during a normal time. If the threshold is trespassed, then in step 2 all the IPs belonging to the same flow and appearing most frequently are disconnected. In step 3, if the connections are below the threshold, stop the disconnection process and the cycle starts over.

Singh and De [129] proposed a method entrenched on an MLP with a Genetic Algorithm (GA) to detect DDoS application layer attacks. Experimental results show that MLP-GA method achieved an accuracy of 98.04% having a rate of 2.21% of False Positive and the use of MLP-GA leads to a better performance than the traditional classifiers e.g., Naive Bayes, RBF Network, MLP, J48, and C45. However, it does not make the distinction between DDoS attacks at the application layer and flash events (which are not attacks). Furthermore, the time-consuming response is not analyzed in the paper. In [155], Araújo et al. made a comparison between the Snort and Suricata IDSs in order to verify which one works better at detecting Slowloris attacks. Snort has been more effective in detecting such attacks, as well as in memory and CPU consumption. However, Suricata was quicker to detect the attack, which means that, although it was fast, it still failed to detect all attacks throughout the experiments.

Muraleedharan and Janet [35] used a deep learning approach to detect Slowloris attacks, which led to an experimental result of 99.61% of accuracy. In [130], Faria et al. developed a tool called SDToW with the aim to detect and block Slowloris attacks in WMNs. Deolindo et al. [238] employed Machine Learning techniques and specifically focused on Quadratic Discriminant Analysis (QDA) as a classification method for IDS to distinguish between legitimate network traffic and malicious attacks. Their research aimed at enhancing the capability of IDS to identify and classify Port Scan and DoS Slowloris attacks more effectively. Asch et al. [241] developed an asynchronous classifier of network flows for detecting Slowloris attacks. This classifier was implemented using random forests and its effectiveness was measured by the Area Under the ROC Curve (AUC). They found that

Table 4.1. Comparison of related works to this article.

Works	Testbed Environment	Network Type	Detection Mechanism(s)	Attack Software	Train/ Test Ratio	Dataset	Performance
Aiello et al. (2013) [124]	LAN	-	SBID	-	-	self-gathered	High anomaly detection, with low probability of false positive rate.
Dantas et al. (2014) [125]	Simulated	Ethernet	SeVen	slowloris.py	-	self-gathered	High availability, greater robustness and less traffic
Mongelli et al. (2015) [126]	LAN	Ethernet	Fast Fourier Transform	-	-	self-gathered	Efficient attack detection by exploiting the flow connection entropy
Katkar et al. (2015) [127]	LAN	-	IDS with Naive Bayesian classifier	slowloris.pl	80/20	self-gathered	Detection accuracy of 97.15%
Aqil et al. (2015) [128]	LAN	Ethernet	High and low traffic threshold	-	-	self-gathered	High TPR and FPR decreased by 66%
Hirakawa et al. (2016) [31]	LAN	Ethernet	Amount of connections per IP addr. and duration time	slowhttptest	-	self-gathered	Highly resistant and and effective detection
Singh and De (2017) [129]	-	-	MLP-GA	slowhttptest	-	public and self-gathered	Detection accuracy of 98.04%
Araújo et. al. (2017) [155]	LAN	Ethernet	Snort and Suricata	slowhttptest	-	self-gathered	Suricata detects less attacks than Snort
Muraleedharan and Janet in (2020) [35]	-	-	Deep learning algorithm	slowhttptest	60/40	public	Detection accuracy of 99.61%
Faria et al. in (2020) [130]	LAN	Wi-fi	SDToW	-	-	self-gathered	Low incidence of false positive errors
Deolindo et al. in (2021) [238]	LAN	Ethernet	QDA	-	60/40	public	Detection accuracy of 98.32%
Oktivasari et al. in (2022) [239]	LAN	Ethernet	iptables	slowloris.py	-	self-gathered	Effective Detection
Murthy et al. in (2023) [240]	LAN	Wi-fi	Thresholds limit and timeout connection by Monte Carlo model	-	-	public and self-gathered	Effective Detection
This article (2023)	Emulated LAN and MAN	Ethernet	ML algorithms, fuzzy logic and Euclidean distance	slowhttptest	75/25	self-gathered	see Section 4.5

using features individually was sufficient to obtain reliable detection results, with two individual features having an AUC greater than 0.95. However, this metric can't be easily compared to the other studies which typically use the accuracy metric.

Murthy et al. in [240] proposed a method to analyze the performance of the Apache2 server using both versions of HTTP, i.e., version 1.1 and 2, in the context of both traditional networks and Software Defined Networks. A threshold is set for parameters such as time to connect to the webpage and latency in obtaining a response from clients using the Monte Carlo model. If the server detects values exceeding these thresholds or if a timeout occurs, it identifies it as malicious activity. Once the attacker's source is pinpointed, the server blocks the client's route, severing the connection between them. Oktivasari et al. in [239] studied the application of iptables as a firewall tool to detect and mitigate the Slowloris attack. The authors emphasize the effectiveness of iptables in managing network traffic, including the ability to limit and block IP addresses directly or completely.

4.3 Methods for Detecting Slowloris Attacks

4.3.1 Classification Features

We manually selected features that lead us to a significant performance during classification without impacting operation in real-time, similarly to other approaches, e.g., [188]. After preliminary experiments with four features, summarized in Figure 4.1, we selected two features to represent the traffic from the HTTP connections: entropy (which captures dispersion well) and the amount of distinct ports. We follow the definition of entropy given by Shannon applied to values for five network traffic features, including a representation of the transport protocol (e.g., TCP, UDP, Internet Control Message Protocol (ICMP)), source and destination IP addresses, and the corresponding port numbers and we also take into consideration the amount of distinct source ports in a given time frame and by transport protocol in the traffic trace.

We use the following method for computing the two features. To a specific trace of traffic, a sliding window of time is applied with a ΔT length according to the period in the Timestamp field for traces marked by the `tshark` tool. For this chapter, we consider $\Delta T = 1$ s by default. Initially, for a 1 s sliding window of time, we assess these two features for all traced traffic packets.

In the first time window, we evaluate the entropy of those packets and the amount of distinct ports found in them. Next, we shift the time window by 1 s, evaluating the amount of distinct ports for the packets and the entropy within this new window. This process is repeated by sliding the time window by $\Delta T = 1$ s up until the end of the trace is reached. The last time window can be a fraction of ΔT, as the duration of the trace may not be a multiple of ΔT. In the next subsections some classification approaches that use the values

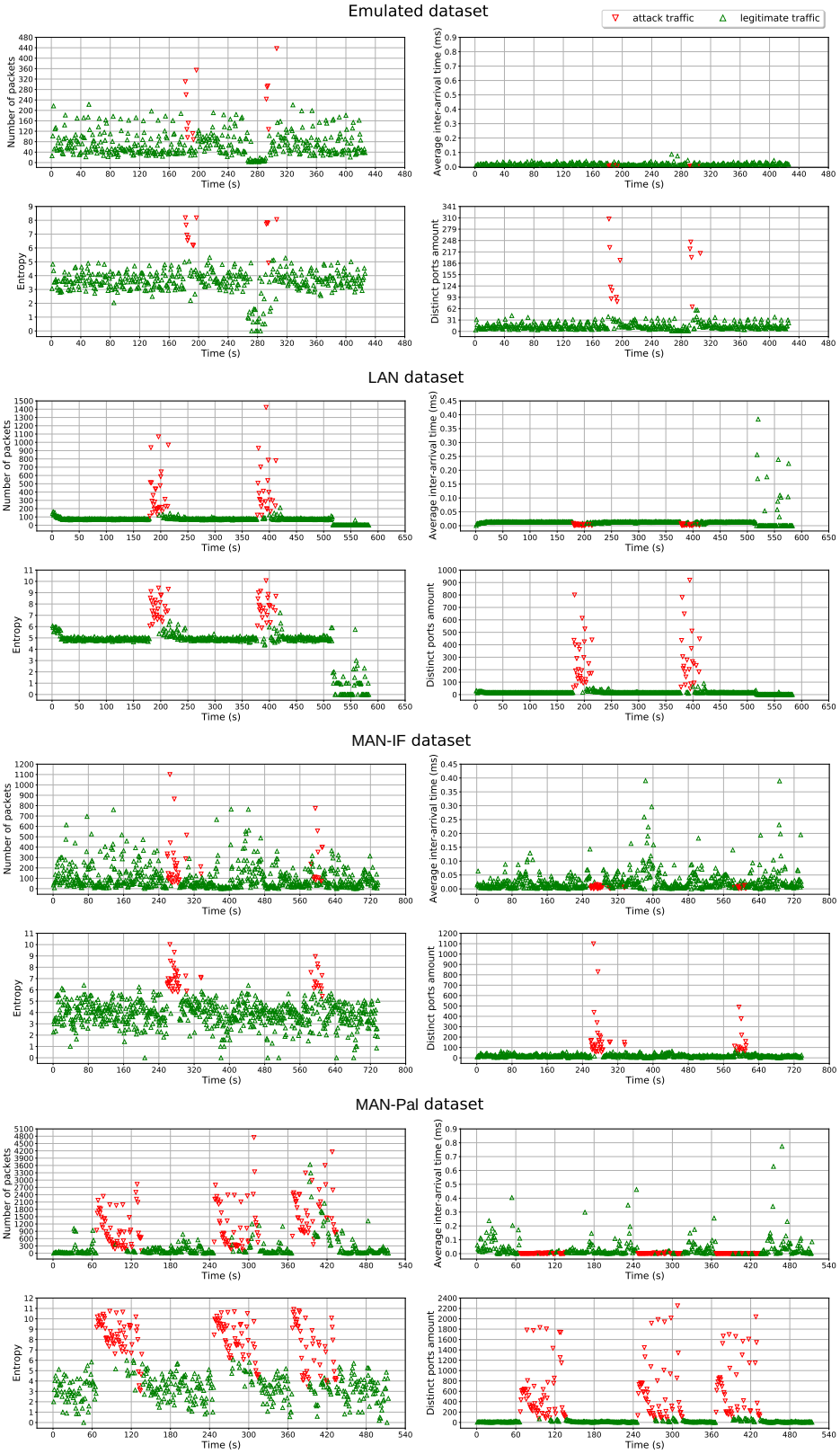

Figure 4.1. Number of packets (top left), Average inter-arrival time (top right), Entropy (bottom left), and Amount of distinct ports (bottom right) for the emulated, LAN, MAN-IF, and MAN-Pal datasets (with a time window of length $\Delta T = 1$ s).

of entropy and amount of distinct ports obtained for each sliding time window to classify the network traffic are described. The set of values of these two features obtained from each sliding window of time is herein called a dataset.

4.3.2 Settings of the Machine Learning Algorithms

To evaluate the feasibility of the selected features and analyze the performance of the nine ML algorithms, we resorted to implementations readily available at *scikit-learn* [223] with their default settings, as summarized in Table 4.2, except for the parameters therein emphasized in bold, which were adjusted according to the result of the `RandomizedSearchCV` algorithm and which are presented here with their default values and in Table 4.3 with their optimized values.

Table 4.2. Hyperparameters' default settings.

Algorithm	Settings
MLP	**activation**=relu, alpha=0.0001, batch_size=auto, beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, **hidden_layer_sizes**=100, **learning_rate**=constant, learning_rate_init=0.001, max_fun=15000, **max_iter**=200, momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5, **random_state**=None, shuffle=True, solver=adam, tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False
K-NN	algorithm=auto, leaf_size=30, metric=minkowski, metric_params=None, n_jobs=None, **n_neighbors**=5, p=2, weights=uniform
SVM	**C**=1, break_ties=False, cache_size=200, class_weight=None, coefo=0.0, decision_function_shape=ovr, degree=3, **gamma**=scale, **kernel**=rbf, max_iter=-1, probability=False, **random_state**=None, shrinking=True, tol=0.001, verbose=False
MNB	**alpha**=1.0, class_prior=None, fit_prior=True
GNB	priors=None, **var_smoothing**=1e-09
DT	ccp_alpha=0.0, class_weight=None, **criterion**=gini, **max_depth**=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, random_state=0, splitter=best
RF	bootstrap=True, ccp_alpha=0.0, class_weight=None, **criterion**=gini, max_depth=None, max_features=auto, max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, **n_estimators**=100, n_jobs=None, oob_score=False, textbfrandom_state=None, verbose=0, warm_start=False
LGBM	bootstrap=True, ccp_alpha=0.0, class_weight=None, criterion=gini, **max_depth**=None, max_features= auto, max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, **n_estimators**=100, n_jobs=None, oob_score=False, **random_state**=None, verbose=0, warm_start=False
XGB	ccp_alpha=0.0, criterion=friedman_mse, init=None, **learning_rate**=0.1, loss=deviance, **max_depth**=3, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, **n_estimators**=100, n_iter_no_change=None, **random_state**=None, subsample=1.0, tol=0.0001, validation_fraction=0.1, verbose=0, warm_start=False

Table 4.3. Hyperparameters' optimized settings.

Algorithm	Parameters
MLP	activation=["identity", "logistic", "tanh", "relu"], hidden_layer_sizes=[100, 200, 300, 400], learning_rate=["constant", "invscaling", "adaptive"], max_iter=[2000, 2500, 3000]
K-NN	n_neighbors=range(1, 50, 1)
SVM	C=[1.0, 10.0, 40.0, 50.0], kernel=["linear", "poly", "rbf", "sigmoid"], gamma=["scale", "auto"]
MNB	alpha=range(1, 1000, 1)
GNB	var_smoothing=np.logspace(0, -9, num=100)
DT	criterion=["gini", "entropy", "log_loss"], max_depth=[2, 10, 15, 50, 100, 200]
RF	criterion=["gini", "entropy", "log_loss"], n_estimators=[100, 200, 300, 400, 500]
LGBM	n_estimators=range(100, 1000, 100), max_depth=range(-1, 50, 1)
XGB	n_estimators=range(100, 1000, 50), learning_rate=np.arange(0.1, 10, 0.1), max_depth=range(3, 100, 3)

The parameters chosen to be adjusted were based on [242, 243, 244]. To adjust them, the randomized search was selected instead of the grid search because, although they explore exactly the same space of parameters and the result in parameter settings is quite similar, the run time for the randomized search is drastically lower [223], which makes it more adequate for this work, leading to the configurations summarized in Table 4.3.

To generate the training and test sets we used the split ratio of 75:25 [245, 246] and the default value of 10 in k-fold for the cross-validation [247]. Among the selected ML al-

gorithms, MLP, RF, SVM, LGBM, DT, and XGB algorithms had the `random_state` field changed to static to avoid the random generation of the values of the features in each iteration round, respectively.

4.3.3 Method Built on Fuzzy Logic, Random Forest and Euclidean Distance

This method derives from a mix of three methods: FL, RF, and ED. First, the traffic is classified in a given trace using the FL method described below. It is next classified (in the same trace) with the RF algorithm, described in the previous subsection, due to the satisfactory classification results of this algorithm (as shown in Section 5).

If both traffic classification results obtained with FL and RF are the same, then that is the classification final result. Otherwise, we perform a further step, that consists in applying ED to obtain, for each value of the amount of distinct ports feature and entropy, the minimum distance. In this latter case, this additional step completes what we call the FRE method. This leads to the two approaches yielding different classifications for every value of the amount of distinct ports and entropy in the dataset used for training. A classification is then obtained using ED for the amount of distinct ports and entropy because the values of the distinct ports amount and entropy in the dataset used for training match known traffic beforehand tagged as legitimate or attack. Finally, we compare the classification of the amount of distinct ports and entropy using the ED with the classification obtained through the fuzzy logic approach. Given that we have two classifications now, when the quantity classified as ”attack type” is higher than the quantity classified as ”legitimate type” classifications, which means that it is an attack and vice versa. When the classification achieved through ED for the amount of distinct ports and the classification achieved through fuzzy logic is not the same, in the two-classification set, it means one classification corresponds to an attack and the other to legitimate, which gives us a warning as the final result. Ensuing, warning alert packets are studied and contrasted to the attack traffic packets that have been blocked. If the two packet sets are alike related to the great values for the amount of distinct ports (distinct ports amount higher than 80), the suspected network traffic is going to be blocked. If not, the destination receives the network packets that pointed to warning alerts.

Regarding FL, we consider a typical fuzzy expert system, where three linguistic terms are used to configure the fuzzy, specifically `low`, `medium` and `high` (for each of the two features entropy and amount of distinct ports). Each linguistic term was chosen based on the amount of traffic a web server can receive. Low traffic concerns situations with low customer access; medium traffic concerns situations caused by some regional event in the institution, resulting in higher activity than usual; and high traffic concerns situations caused by some kind of DDoS attack or a flash crowd type of event. The ranges of values for each linguistic term were based on the average values generated for each attribute per

dataset, as shown in Figure 4.2.

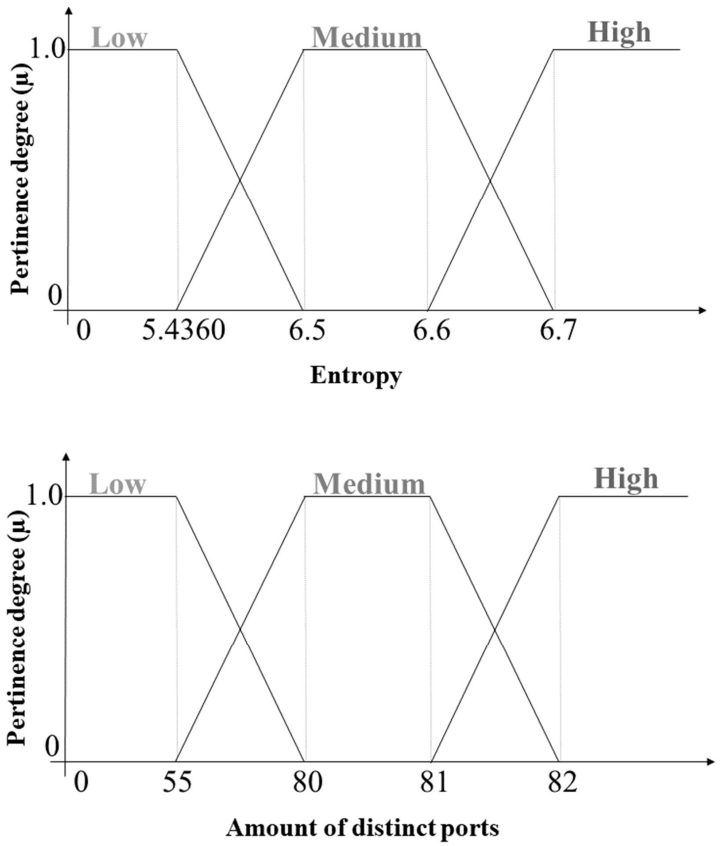

Figure 4.2. Pertinence function for the features entropy and amount of distinct ports.

The pertinence function that was chosen to normalize the values of the data is, for every feature, trapezoidal. The selected value range for the linguistic terms in the universe of discourse for every single variable is derived from the datasets, more specifically from the values of the two evaluated features from it. Consequently, the `low` term is used for the lowest values of the features, `high` for the highest values, and `medium` for the average ones. Following fuzzification, the fuzzy rules set receives the degree of pertinence values created by each input variable. It is made up of 9 rules, that must cover all possible scenarios for system fuzzy behaviour, as shown in Figure 4.3. The Mamdani inference model [225] is applied in all the rules.

RULE	IF (ANTECEDENT)						THEN (CONSEQUENT)
	AMOUNT OF DISTINCT PORTS			ENTROPY			CLASSIFICATION
	LOW	MEDIUM	HIGH	LOW	MEDIUM	HIGH	
1			⊗			⊗	ATTACK
2			⊗		⊗		ATTACK
3			⊗	⊗			ATTACK
4		⊗				⊗	ATTACK
5		⊗			⊗		ATTACK
6		⊗		⊗			ATTACK
7	⊗					⊗	ATTACK
8	⊗				⊗		LEGITIMATE
9	⊗			⊗			LEGITIMATE

Figure 4.3. Base rules for the inference module.

The AND operator is applied on the predecessors of every rule, and the lowest value is kept among the triggered rule pertinence degree values, as a consequence. When the inference module has triggered every single rule, the defuzzification starts. To translate the pertinence degree values into an output providing accurate numerical values, we use the Center-of-Area (CoA) method [50] to defuzzify them. The resulting output is then categorized according to the linguistic terms *"legitimate"* or *"attack"*.

4.4 Experimental Environments and Datasets

Due to the lack of publicly available distributed Slowloris attack traffic traces, we created such traces in three distinct environments: emulated, local (LAN) area, and metropolitan (MAN) area environments. The emulated environment was created with four attackers, three normal clients, and one web server, using the *Netkit* [227] software in all emulated computers and network devices.

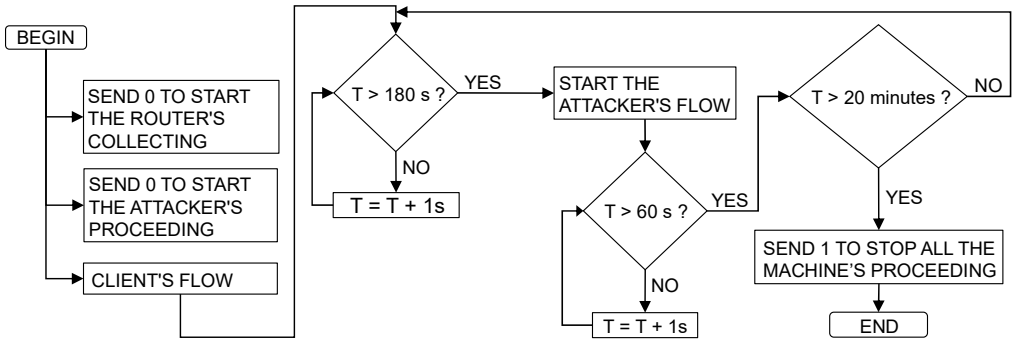

Figure 4.4. Flowchart of the attack mimicking procedure.

The local environment was created with five attackers and one web server at IFTO in Brazil, which has a network infrastructure with routers distributed among different buildings, that allow data traffic coming from different locations. We created two metropolitan area environments with the same architecture, but with different web server locations: one with the web server located at IFTO (MAN-IF) and another one with the web server located at the Palmas municipal government (MAN-Pal). Each metropolitan environment was created with four attackers distributed around the city of Palmas (Brazil) using the available Internet infrastructure to attack the web server located at IFTO (MAN-IF) or located at Palmas municipal government (MAN-Pal). Normal clients in LAN and MAN environments are the users accessing those servers. The security assessments mimicking real attacks on web servers of IFTO and Palmas municipal government were carried out for research purposes only, with the required institutional authorizations and in compliance with the values of academic integrity and code of ethics. The attack procedure follows the flowchart shown in Figure 4.4 for the three environments. Each attack-mimicking process takes 20 minutes, with a continuous collection of normal client traffic. The attack is initiated 180 s after the beginning and lasts for 60 s. The traffic traces obtained have the

following sizes: 40.6 MB (emulated), 1.0 GB (LAN), 165.51 MB (MAN-IF), and 23.6 MB (MAN-Pal). These four datasets are open data and are freely available for the research community on the CERN's Zenodo platform [248].

We merged the above four traffic traces (emulated, LAN, MAN-IF, and MAN-Pal) into just one to facilitate the application of the 10-fold cross-validation, since this larger dataset is randomly partitioned in a ratio of 75%/25%, in each of the 10 folds (75% is used for training and 25% is used for testing).

4.5 Performance Evaluation

This section offers an assessment of the approaches used for classifying the traces of traffic listed above. We employ the metrics *Precision*, *Recall*, *F1-Score* and *Accuracy* (hereby referred to as *Test accuracy* and *Train accuracy*) as typically described in the literature (e.g., [37]). For some ML algorithms, we also normalized the dataset so that values range from 0 and 1, in accordance with the respective input definition, though we do not specifically mention this afterwards in the chapter.

As explained at the end of Section 4, to apply the 10-fold cross-validation with a split ratio of 75:25 for training and testing, we merged the four datasets into just one. The values shown in Table 4.4 are thus the average of the classification results for both the nine ML algorithms and the FRE method in the different portions of the dataset used for training and for testing in the 10 iterations.

The *Train accuracy* and *Standard deviation* metrics were generated through the *cross_val_score* function with a k-fold of size 10. The value of the *Train accuracy* metrics is an average of the averages produced from 10 runs generated by the *cross_val_score* function for each one of the 10 portions of the dataset randomly generated from the single dataset mentioned above. The same occurs with the *Standard deviation* metric, which is generated based on the *Train accuracy* metric as the result of the *cross_val_score* function, but its value is the average of the 10 portions of the dataset.

The confusion matrix related to the traffic type (i.e., attack or legitimate), has been computed for each algorithm and for the FRE method. It contains the average absolute values of true positives (detected attacks) on the upper left corner, true negatives (regular traffic) on the lower right, false positives (false attacks) on the lower left corner and false negatives (undetected attacks) on the upper right corner.

4.5.1 Classification Results

Table 4.4 summarizes the results of the classification approaches using both default and optimized values or the hyperparameters. The values in the table in plain display were

Table 4.4. Classification results (sorted by Test accuracy %) for the default hyperparameters and for the optimized hyperparameters (given as a delta in square brackets when $\neq 0$). N: Normalized data.

Algorithm	**Traffic**	**Precision** in %	**Recall** in %	**F1-Score** in %	**Confusion matrix** *attack*	**Confusion matrix** *legitimate*	**Test accuracy** in %	**Train accuracy** in %	**Standard deviation**
MNB	*attack*	93.47[-1.10]	89.25[+5.52]	91.30[+2.25]	TP=59.80[+3.70]	FN=4.20[+1.10]	*97.98[+0.46]*	*97.79[+0.46]*	*0.011508[-0.001998]*
	legitimate	98.56[+0.73]	99.16[-0.22]	98.86[+0.26]	FP=7.20[-3.70]	TN=493.80[-1.10]			
SVM(N)	*attack*	99.01[-3.29]	89.70[+5.97]	94.10[+1.56]	60.10[+4.00]	0.60[+2.30]	*98.67[+0.30]*	*98.53[+0.76]*	*0.007353[-0.001289]*
	legitimate	98.63[+0.79]	99.88[-0.46]	99.25[+0.17]	6.90[-4.00]	497.40[-2.30]			
GNB	*attack*	93.31[+1.21]	99.70[-1.34]	96.36	66.80[-0.90]	4.90[-1.00]	*99.10[+0.02]*	*99.07*	*0.007501*
	legitimate	99.96[-0.18]	99.02[+0.20]	99.48[+0.01]	0.20[+0.90]	493.10[+1.00]			
MLP(N)	*attack*	98.61[+0.17]	94.93[+0.90]	96.72[+0.54]	63.60[+0.60]	0.90[-0.10]	*99.24[+0.12]*	*99.10[+0.19]*	*0.006776[-0.000852]*
	legitimate	99.32[+0.12]	99.82[+0.02]	99.57[+0.07]	3.40[-0.60]	497.10[+0.10]			
DT	*attack*	96.79[+0.43]	97.31[+0.30]	97.04[+0.36]	65.20[+0.20]	2.20[-0.30]	*99.29[+0.09]*	*99.10[+0.11]*	*0.006886[-0.000591]*
	legitimate	99.64[+0.04]	99.56[+0.06]	99.60[+0.05]	1.80[-0.20]	495.80[+0.30]			
K-NN(N)	*attack*	97.35[+0.17]	97.16[-0.15]	97.24[-0.01]	65.10[-0.10]	1.80[-0.10]	*99.35*	*99.23[-0.17]*	*0.006700[+0.000813]*
	legitimate	99.62[-0.02]	99.64[+0.02]	99.63	1.90[+0.10]	496.20[+0.10]			
XGB	*attack*	97.39[+0.42]	98.36[-0.30]	97.85[+0.07]	65.90[-0.20]	1.80[-0.30]	*99.49[+0.02]*	*99.33[-0.23]*	*0.006253[+0.000652]*
	legitimate	99.78[-0.04]	99.64[+0.06]	99.71[+0.01]	1.10[+0.20]	496.20[+0.30]			
LGBM	*attack*	97.66[-0.29]	98.36[+0.15]	98.00[-0.07]	65.90[+0.10]	1.60[+0.20]	*99.52[-0.02]*	*99.36*	*0.006494*
	legitimate	99.78[+0.02]	99.68[-0.04]	99.73[-0.01]	1.10[-0.10]	496.40[-0.20]			
RF	*attack*	97.25[+0.28]	98.81[-0.15]	98.01[+0.07]	66.20[-0.10]	1.90[-0.20]	*99.52[+0.02]*	*99.37[-0.01]*	*0.006124[-0.000166]*
	legitimate	99.84[-0.02]	99.62[+0.04]	99.73[+0.01]	0.80[+0.10]	496.10[+0.20]			
FRE	*attack*	98.68	99.70	99.18	67.40[-0.10]	0.20	*99.80*	*99.37[-0.01]*	*0.006124[-0.000166]*
	legitimate	99.96	99.82	99.89	0.90	494.90[+0.30]			

obtained for the default values of the hyperparameters; in square brackets we present the differences of the results to the ones outputted for the optimized hyperparameters.

The table highlights that, for the nine ML algorithms with default parameters, *Precision* ranges from 93.47% to 99.01% for the attack traffic. The legitimate traffic is also properly classified with values ranging from 98.56% to 99.96%, with two algorithms above 99.80%. *Test accuracy* ranges from 97.98% to 99.52% and *Train accuracy* ranges from 97.79% to 99.37%, which shows that these ML algorithms are efficient when applied to Slowloris attack detection.

Results in Table 4.4 prove that most of the ML algorithms are generalizing well. The *Standard deviation* shows that the training accuracy of the algorithms has a low dispersion on all the iterations, except for the MNB algorithm with default hyperparameters, which has a slightly worse deviation value than the others. In both default and optimized cases, the FRE method and the RF algorithm have the same values for the *Train accuracy* and *Standard deviation* metrics. This is due to the inclusion of the RF algorithm in the FRE method.

Table 4.5. Warning alerts - in parentheses if only found with default hyperparameters, in square brackets if only found with optimized hyperparameters.

Dataset round	# of distinct ports	Entropy	ED port classif.	ED entropy classif.	FL classif.	RF classif.
1	63	5.2937	attack	legitimate	attack	legitimate
	54	6.3059	legitimate	attack	legitimate	attack
	53	6.5016	legitimate	attack	legitimate	attack
2	67	5.1669	attack	legitimate	attack	legitimate
	59	3.4315	attack	legitimate	attack	legitimate
	[78]	[5.8722]	attack	legitimate	attack	legitimate
3	(52)	(6.5016)	legitimate	attack	legitimate	attack
	65	5.9644	legitimate	attack	attack	legitimate
4	71	3.8757	attack	legitimate	attack	legitimate
5	56	5.8129	legitimate	attack	attack	legitimate
	53	6.5016	legitimate	attack	legitimate	attack
6	76	4.9252	attack	legitimate	attack	legitimate
	(53)	(6.5016)	legitimate	attack	legitimate	attack
	54	6.3059	legitimate	attack	legitimate	attack
	63	5.2937	attack	legitimate	attack	legitimate
7	(65)	(5.9644)	legitimate	attack	attack	legitimate
	78	3.7911	attack	legitimate	attack	legitimate
	76	2.8301	attack	legitimate	attack	legitimate
	78	2.8301	attack	legitimate	attack	legitimate
8	63	5.2937	attack	legitimate	attack	legitimate
	59	3.4357	attack	legitimate	attack	legitimate
	53	6.4621	legitimate	attack	legitimate	attack
9	56	4.8959	attack	legitimate	legitimate	attack
	57	4.7329	attack	legitimate	attack	legitimate
10	78	5.8722	attack	legitimate	attack	legitimate
	71	3.8741	attack	legitimate	attack	legitimate
	66	4.9212	attack	legitimate	attack	legitimate
	68	5.1669	attack	legitimate	attack	legitimate

The FRE method with default parameters reaches a *Test accuracy* of 99.80% and a *Train accuracy* of 99.37%. These values are higher than the ones obtained for the ML algorithms as well as the ones found in the literature (when available) as show in Table 1. The FRE method scores at 98.68% for *Precision*, 99.70% for *Recall*, and 99.18% for *F1-Score* on

the attack traffic. These high scores are also reflected in the confusion matrix of the FRE method where the number of false negatives and false positives are extremely low.

The traffic marked as *"warning"* is analyzed and, if it has the same characteristics as the attack traffic, it will be blocked; otherwise, it will be allowed to proceed to the destination. As can be seen in Table 4.5, the features *amount of distinct ports* and *entropy* have similar values between legitimate and attack traffic. This can happen because a sliding window with a length of 1 s can contain a lot of legitimate traffic from different sources causing a high value for this feature, sometimes leading to a high rate of false positives. Therefore, in the dataset, the traffic with no attack traffic but with these values was classified as legitimate, and vice versa.

Contextualizing our results with the related work, and although the datasets are different (and therefore the results are not directly comparable), the FRE method led, as far as we know, to the best classification results for Slowloris attacks achieved so far.

4.5.2 Resource Usage

A computer with a quad-core 64-bit processor, 8 GB of RAM, and running Ubuntu 20.04 was used to perform the experiments reported herein and obtain values related to the computational resources consumed by the FRE method as well as the ML algorithms. Table 4.6 displays the resources used by each classifier during the classification process that led to the results presented in Table 4.4. The depicted values are an average of 10 execution rounds for each algorithm and for the FRE method. CPU usage, RAM usage, and execution times were measured with the GNU `time` command. We can see that CPU usage is above 100.00%, meaning that more than 1 core was required for classifying the entire dataset. The algorithms are sorted from the one consuming the least CPU to the one using the most CPU, when using default hyperparameters.

As can be seen in Table 4.6, the FRE method consumes less CPU than all ML algorithms for hyperparameters with default values. It is the same for hyperparameters with optimized values, except for the RF and XGB algorithms which have a few % lower usage. It also consumes less RAM than the LGBM algorithm for hyperparameters with default and optimized values. However, compared to the other algorithms, it consumes around one-third more RAM with default hyperparameters and one-tenth with optimized hyperparameters. The superior performance of the FRE method has a cost in terms of execution time, only being less costly than the MLP algorithm and the LGBM algorithm with optimized values. In order to perform the classification, FRE requires 0.69 s when using default hyperparameters and 0.8 s when using optimized hyperparameters. Meanwhile, the other ML algorithms require from 0.37 s to 1.07 s for both types of hyperparameters, except for the MLP algorithm, which is the slowest.

The FRE method shows the best classification results compared to the ML algorithms but

Table 4.6. Computational resource usage (sorted by CPU%).

Algorithm	Default parameters	Optimized parameters
FRE	CPU: 137.10% RAM: 112.85MB Execution time: 0.69 s	CPU: 148.10% RAM: 100.56MB Execution time: 0.80 s
LGBM	CPU: 137.90% RAM: 115.78MB Execution time: 0.63 s	CPU: 148.10% RAM: 130.55MB Execution time: 1.07 s
XGB	CPU: 138.40% RAM: 88.98MB Execution time: 0.54 s	CPU: 143.50% RAM: 98.96MB Execution time: 0.73 s
RF	CPU: 139.30% RAM: 89.16MB Execution time: 0.54 s	CPU: 145.10% RAM: 99.36MB Execution time: 0.79 s
SVM	CPU: 153.70% RAM: 82.88MB Execution time: 0.41 s	CPU: 172.20% RAM: 92.80MB Execution time: 0.46 s
K-NN	CPU: 154.10% RAM: 84.62MB Execution time: 0.41 s	CPU: 170.20% RAM: 94.20MB Execution time: 0.49 s
DT	CPU: 156.50% RAM: 85.97MB Execution time: 0.40 s	CPU: 159.30% RAM: 95.62MB Execution time: 0.49 s
GNB	CPU: 159.40% RAM: 77.73MB Execution time: 0.37 s	CPU: 175.60% RAM: 86.52MB Execution time: 0.43 s
MNB	CPU: 160.60% RAM: 78.23MB Execution time: 0.38 s	CPU: 179.60% RAM: 86.95MB Execution time: 0.42 s
MLP	CPU: 246.30% RAM: 79.72MB Execution time: 1.73 s	CPU: 282.30% RAM: 90.01MB Execution time: 3.01 s

it also requires more RAM and more execution time. However, those resources remain in the same order of magnitude than the ones used by the ML algorithms. Therefore, the FRE method is perfectly suitable for online operation. Indeed, looking at both tables 4.4 and 4.6, we can observe that the algorithms can be roughly placed in three categories: (i) the first category has a fast execution time but a low accuracy and includes the GNB, MNB, and SVM algorithms; (ii) the second category has both an average execution time and an average accuracy and includes the K-NN, DT, and XGB algorithms; and (iii), the third category has a slow execution time and a high accuracy and includes the LGBM, RF and FRE algorithms. MLP is off-category with a high CPU usage and a very high execution time despite a reasonable accuracy. Therefore, a trade-off can be made depending on the requirements of the detection system regarding real-time processing and traffic volume.

4.6 Conclusion

We assessed and compared nine ML algorithms as well as a combined method with three algorithms (Fuzzy Logic, Random Forest, and Euclidean Distance) for detecting distributed Slowloris attacks. Those algorithms and the FRE method were evaluated using a union of

emulated, LAN, and MAN traffic traces. For both the hyperparameters with default values and the hyperparameters with optimized values, the best classification results were obtained by our FRE method with a *Test accuracy* of 99.8%, although the gap was thin with some other ML algorithms such as the Random Forest and LGBM. Indeed, among the nine ML algorithms evaluated, five of them (DT, K-NN, XGB, LGBM, and RF) had all their scores above 95% which makes them appropriate candidates for the task. However, only our FRE method had all its scores above 98.5%.

In most of the cases, using the optimized hyperparameters only made a small improvement to the results obtained by using the default hyperparameters. This can ease the work of practitioners who will not be required to optimize the hyperparameters before use if they can cope with a small loss of accuracy. We showed that the algorithms exhibit a trade-off between speed and accuracy, and could be selected depending on the requirements of the detection system. Regarding the resource usage, FRE is the least CPU-intensive algorithm but is on the upper side regarding the RAM consumption and the execution time. The latter still being totally compatible with an online operation mode for a detection system.

For future work, we intend to compare our proposal with unsupervised machine learning algorithms, measuring the classification accuracy by the attributes chosen by the algorithms for the detection of Slowloris attacks. In addition, we also plan to generate new datasets with a huge volume of data traffic produced by new environments with the addition of many endpoint and intermediate devices, expanding the paths on which the attack traffic travels to reach the target. In this way, new values of the attributes will be generated, which could be used for the detection of such attacks.

Chapter 5

Conclusions and Future Work

This Chapter presents the main conclusions that result from the research work described in this thesis. Furthermore, it discusses a few research topics related with the work developed in the doctoral program that may be addressed in the future.

5.1 Main Conclusions

DoS attacks have evolved over the years targeting different vulnerabilities with the main purpose of blocking the resources allocated to the services. In the first and second generations of attacks, carried out by a single computer or by a set of distributed computers, respectively, the common denominator was the huge amount of traffic sent to the victim. In the third generation, attacks became stealthy and changed the way the amount of traffic is sent to the victim. They are now low-rate traffic attacks. This new type of attack became a big challenge to be addressed by the most recent detection approaches.

This thesis provides a comprehensive study of LDoS attacks, categorizing them according to format, target, and objective. It also discusses detection mechanisms that identify attack fingerprints in network traffic and presents tools capable of replicating LDoS attack patterns. Next, this thesis focuses on detecting two types of LDoS attacks: RoQ and Slowloris. For this purpose, two approaches have been developed: one approach is based on the separate use of a set of traditional machine learning algorithms and the second approach is based on fuzzy logic plus one traditional machine learning algorithm (that previously led to good classification results) and Euclidean distance.

For the proposed RoQ attack detection framework, the lack of a publicly available database containing RoQ attack traffic was one of the most significant challenges during this research. Consequently, test environments had to be set up to generate datasets containing RoQ attack traffic. To evaluate the performance of the detection algorithms, two different scenarios were selected: Emulated and LAN environments. The proposed approach, which integrates fuzzy logic, neural networks, and Euclidean distance (FME), achieved a higher classification accuracy (99.36% in the emulated environment and 100.00% in the LAN environment) compared to the approach based on the separate use of 11 machine learning algorithms. However, in terms of resource usage, our approach exhibited a longer execution time, primarily due to the computational demands of the neural network algorithm and the additional overhead introduced by the integrated methodology.

Nonetheless, machine learning algorithms, including our approach, have demonstrated strong generalization capabilities when applied to datasets from different domains.

Regarding the proposed Slowloris attack detection framework, the lack of a publicly available database containing distributed Slowloris attack traffic was also one of the most significant challenges during this research. Therefore, test environments had to be set up to generate datasets containing Slowloris attack traffic. Four different scenarios were selected to evaluate the performance of the detection algorithms: Emulated, LAN, MAN-IF, and MAN-PAL environments. The proposed approach, integrating Fuzzy Logic, Random Forest, and Euclidean Distance (FRE), achieved the best classification performance, reaching 99.80% accuracy in traffic classification compared to the nine machine learning algorithms considered here. Concerning resource usage, our approach exhibited larger RAM consumption and execution time, mainly due to the computational overhead introduced by the integrated methodology. Furthermore, despite the hyperparameter optimization, the classification algorithms did not achieve a significant improvement compared to their default settings.

5.2 Future Work

The lack of datasets containing distributed low-rate DoS attack traffic will be the major problem to be addressed in future works. Although there are sites that provide DDoS datasets, most of them are either high-rate traffic, or generic low-rate attack traffic, or single Slow DoS attack traffic. Therefore, more datasets with specific distributed low-rate and Slow DoS attack traffic are needed to evaluate the accuracy of the approaches that use artificial intelligent methods.

The development of a tool to detect these attacks in real-time is also a worthwhile objective. Detection time is still a non-negligible factor in the current proposal. Indeed, due to the randomness of the flood traffic mixed with other legitimate traffic and other things related to the traffic in production environments, it is still difficult to speed up the detection process. The selection of new features could contribute to improve the foremost detection performance metric which involves distinguishing the attack traffic from the legitimate traffic.

New artificial intelligence methods, such as deep learning, could be leveraged to increase classification performance and/or reduce the usage of computational resources. Accurate evaluations regarding classification time and correctness provided by these new methods should be investigated.

Bibliography

[1] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants," *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*," New York, NY, United States. ACM, 2003. doi: 10.1145/863955.863966 pp. 75–86. xxv, 12, 17, 49

[2] A. Shevtekar and N. Ansari, "A router-based technique to mitigate reduction of quality (RoQ) attacks," *Computer Networks*, vol. 52, no. 5, pp. 957–970, 2008. doi: 10.1016/j.comnet.2007.11.015 xxv, 17, 18, 35, 37, 52, 53

[3] C. Xu, J. Shen, and X. Du, "Low-rate DoS attack detection method based on hybrid deep neural networks," *Journal of Information Security and Applications*, vol. 60, p. 102879, 2021. doi: 10.1016/j.jisa.2021.102879 xxv, 17

[4] S. Tayama and H. Tanaka, "Analysis of Slow Read DoS Attack and Communication Environment," *ICMWT 2017: International Conference on Mobile and Wireless Technologies 2017*," Singapore, Malaysia. Springer, 2017. doi: 10.1007/978-981-10-5281-1_38 pp. 350–359. xxv, 20

[5] E. Cambiaso, G. Chiola, and M. Aiello, "Introducing the SlowDrop Attack," *Computer Networks*, vol. 150, pp. 234–249, 2019. doi: https://doi.org/10.1016/j.comnet.2019.01.007 xxv, 21, 22, 23, 49

[6] G. Maciá-Fernández, J. E. Díaz-Verdejo, and P. García-Teodoro, "Evaluation of a low-rate DoS attack against iterative servers," *Computer networks*, vol. 51, no. 4, pp. 1013–1030, 2007. doi: 10.1016/j.comnet.2006.07.002 xxv, 23, 24, 49

[7] G. Maciá-Fernández, J. E. Díaz-Verdejo, and P. García-Teodoro, "Evaluation of a low-rate DoS attack against application servers," *Computers & Security*, vol. 27, no. 7, pp. 335–354, 2008. doi: 10.1016/j.cose.2008.07.004 xxv, 24

[8] H. S. Mondal, M. T. Hasan, M. B. Hossain, M. E. Rahaman, and R. Hasan, "Enhancing Secure Cloud Computing Environment by Detecting DDoS Attack Using Fuzzy Logic," *2017 3rd International Conference on Electrical Information and Communication Technology (EICT)*," Khulna, Bangladesh. IEEE, 2017. doi: 10.1109/EICT.2017.8275211 pp. 1–4. xxv, 51, 58, 60

[9] Statista, "E-commerce worldwide - Statistics Facts. Accessed at: 19 out. 2022," 2021. [Online]. Available: https://www.statista.com/topics/871/online-shopping/#dossier-chapter1 1

[10] e-commercebrasil excelência em ecommerce, "A influência do mobile e o papel do e-commerce no crescimento econômico. Accessed at: 14 out.

2022," 2019. [Online]. Available: https://www.ecommercebrasil.com.br/artigos/influencia-mobile-e-commerce/ 1

[11] V. S.A., "Compras pelo celular representam 55,1% das vendas online em 2020. Accessed at: 29 out. 2022," 2021. [Online]. Available: https://cndl.org.br/varejosa/compras-pelo-celular-representam-551-das-vendas-online-em-2020/ 1

[12] ERICSSON, "Mobile data traffic outlook. Accessed at: 10 may 2024," 2024. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast 1

[13] W. conectologia, "Ataques cibernéticos – o que são e como se proteger? Accessed at: 10 set. 2022," 2021. [Online]. Available: https://wcsconectologia.com.br/blog/ataques-ciberneticos-o-que-sao-e-como-se-proteger/ 1

[14] C. Brasil, "Ataques cibernéticos a empresas brasileiras crescem 220% no 1º semestre de 2021. Accessed at: 10 set. 2022," 2021. [Online]. Available: https://www.cnnbrasil.com.br/business/ataques-ciberneticos-a-empresas-brasileiras-crescem-220-no-1-semestre-de-2021/ 1

[15] E. S. Report, "C183R536UR4NÇ4 - Conheça os desafios para as empresas da América Latina. Accessed at: 10 set. 2022," 2021. [Online]. Available: https://encurtador.com.br/fjoDF 1

[16] E. C. Design, "The History and Evolution of DDoS Attacks. Accessed at: 11 set. 2022," 2020. [Online]. Available: https://www.embeddedcomputing.com/technology/security/network-security/the-history-and-evolution-of-ddos-attacks 1

[17] M. Azure, "Azure DDoS Protection—2021 Q1 and Q2 DDoS attack trends. Accessed at: 11 set. 2022," 2021. [Online]. Available: https://azure.microsoft.com/en-us/blog/azure-ddos-protection-2021-q1-and-q2-ddos-attack-trends/ 1

[18] Akamai, "A Retrospective on DDoS Trends in 2023 and Actionable Strategies for 2024. Accessed at: 10 feb. 2024," 2024. [Online]. Available: https://www.akamai.com/blog/security/a-retrospective-on-ddos-trends-in-2023 2

[19] A. Kazmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks," *Proceedings of Symposium, Communication Architecture Protocol*," Karlesruhe, Germany, 2003, pp. 345–350. 2

[20] W. Zhijun, L. Wenjing, L. Liang, and Y. Meng, "Low-rate DoS attacks, detection, defense, and challenges: a survey," *IEEE Access*, vol. 8, pp. 43 920–43 943, 2020. doi: 10.1109/ACCESS.2020.2976609 2, 14, 15

[21] E. Cambiaso, G. Papaleo, G. Chiola, and M. Aiello, "Slow DoS attacks: definition and categorisation," *International Journal of Trust Management in Computing and Communications*, vol. 1, no. 3-4, pp. 300–319, 2013. doi: 10.1504/IJTMCC.2013.056440 2, 13, 15

[22] Q. Zhu, Z. Yizhi, and X. Chuiyi, "Research and Survey of Low-rate Denial of Service Attacks," *13th International Conference on Advanced Communication Technology (ICACT2011)*," Gangwon, Korea (South). IEEE, 2011, pp. 1195–1198. 2, 12, 13, 15

[23] R. Mathew and V. Katkar, "Survey of Low Rate DoS Attack Detection Mechanisms," *ICWET '11: Proceedings of the International Conference Workshop on Emerging Trends in Technology*," New York, NY, United States. ACM, 2011. doi: 10.1145/1980022.1980227 pp. 955–958. 2, 13, 15

[24] X. Liu, G. Cheng, Q. Li, and M. Zhang, "A comparative study on flood DoS and low-rate DoS attacks," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, pp. 116–121, 2012. doi: 10.1016/S1005-8885(11)60458-5 2, 12, 13, 15

[25] L. Mohan, M. Bijesh, and J. K. John, "Survey of Low rate Denial of Service (LDoS) attack on RED and its counter strategies," *2012 IEEE International Conference on Computational Intelligence and Computing Research*," Coimbatore, India, 2012. doi: 10.1109/ICCIC.2012.6510186 pp. 1–7. 2, 13, 15

[26] "Taxonomy of Slow DoS Attacks to Web Applications," *Recent Trends in Computer Networks and Distributed Systems Security*, vol. 335, pp. 195–204, 2012. doi: 10.1007/978-3-642-34135-9_20 2, 13, 15, 19, 23

[27] N. Tripathi and N. Hubballi, "Application Layer Denial-of-Service Attacks and Defense Mechanisms: A Survey," *ACM Computing Surveys*, vol. 54, no. 4, 2021. doi: 10.1145/3448291 2, 14, 15

[28] D. Zhang, Q. Wang, G. Feng, Y. Shi, and A. V. Vasilakos, "A survey on attack detection, estimation and control of industrial cyber–physical systems," *ISA Transactions*, vol. 116, pp. 1–16, 2021. doi: 10.1016/j.isatra.2021.01.036 2, 14

[29] V. de Miranda Rios, P. R. M. Inácio, D. Magoni, and M. M. Freire, "Detection and Mitigation of Low-Rate Denial-of-Service Attacks: A Survey," *IEEE Access*, vol. 10, pp. 76 648–76 668, 2022. doi: 10.1109/ACCESS.2022.3191430 2, 7, 8, 11

[30] M. Guirguis, A. Bestavros, and I. Matta, "Exploiting the transients of adaptation for RoQ attacks on Internet resources," *International Conference on Network Protocols*," Berlin, Germany, 2004. doi: 10.1109/ICNP.2004.1348109 pp. 184–195. 3, 18, 23, 49, 50

[31] T. Hirakawa, K. Ogura, B. B. Bista, and T. Takata, "A defense method against distributed slow HTTP DoS attack," *2016 19th International Conference on Network-Based Information Systems (NBiS)*," Ostrava, Czech Republic. IEEE, 2016. doi: 10.1109/NBiS.2016.58 pp. 152–158. 3, 39, 40, 81, 83, 84

[32] V. de Miranda Rios, P. R. M. Inácio, D. Magoni, and M. M. Freire, "Detection of Slowloris Attacks using Machine Learning Algorithms," *The 39th ACM/SIGAPP Symposium on Applied Computing (SAC'24)*," Avila, Spain. ACM, 2024. doi: 10.1145/3605098.3635919 pp. 1320–1329. 3, 7, 8, 9, 81

[33] M. M. Hamdi, S. A. Rashid, M. Ismail, M. A. Altahrawi, M. F. Mansor, and M. K. AbuFoul, "Performance evaluation of active queue management algorithms in large network," *2018 IEEE 4th International Symposium on Telecommunication Technologies (ISTT)*," Selangor, Malaysia. IEEE, 2018. doi: 10.1109/ISTT.2018.8701716 pp. 1–6. 3

[34] Y. Chen and K. Hwang, "Spectral analysis of TCP flows for defense against reduction-of-quality attacks," *2007 IEEE International Conference on Communications*," Glasgow, UK. IEEE, 2007. doi: 10.1109/ICC.2007.204 pp. 1203–1210. 4, 35, 37, 52, 53

[35] N. Muraleedharan and B. Janet, "A deep learning based HTTP slow DoS classification approach using flow data," *ICT Express*, vol. 7, no. 2, pp. 210–214, 2021. doi: 10.1016/j.icte.2020.08.005 4, 83, 84

[36] M. A. R. Ken Peffers, Tuure Tuunanen and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007. doi: 10.2753/MIS0742-1222240302 6

[37] V. de Miranda Rios, P. R. M. Inácio, D. Magoni, and M. M. Freire, "Detection of reduction-of-quality DDoS attacks using Fuzzy Logic and machine learning algorithms," *Computer Networks*, vol. 186, pp. 1–18, 2021. doi: 10.1016/j.comnet.2020.107792 7, 8, 37, 38, 44, 47, 81, 91

[38] SANS, "The changing face of distributed denial of service mitigation. Accessed at: 24 fev. 2022," 2001. [Online]. Available: https://www.sans.org/white-papers/462/ 12

[39] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Computing Surveys*, vol. 39, no. 1, p. 3, 2007. doi: 10.1145/1216370.1216373 12, 23, 48

[40] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643–666, 2003. doi: 10.1016/j.comnet.2003.10.003 12, 48

[41] R. K. C. Chang, "Defending against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 42–51, 2002. doi: 10.1109/MCOM.2002.1039856 12, 48

[42] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004. doi: 10.1145/997150.997156 12, 48

[43] H. Beitollahi and G. Deconinck, "Analyzing well-known countermeasures against distributed denial of service attacks," *Computer Communications*, vol. 35, no. 11, pp. 1312–1332, 2012. doi: 10.1016/j.comcom.2012.04.008 12

[44] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, 2013. doi: 10.1016/j.comnet.2012.07.021 12

[45] S. T. Zargar, J. Joshi, and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013. doi: 10.1109/SURV.2013.031413.00127 12, 49

[46] L. Garber, "Denial-of-service Attacks Rip the Internet," *IEEE Computer Society*, vol. 33, no. 4, pp. 12–17, 2000. doi: 10.1109/MC.2000.839316 12

[47] G. Maciá-Fernández, R. A. Rodríguez-Gómez, and J. E. Díaz-Verdejo, "Defense techniques for low-rate DoS attacks against application servers," *Computer Networks*, vol. 54, no. 15, pp. 2711–2727, 2010. doi: 10.1016/j.comnet.2010.05.002 12, 42, 43

[48] Z. Wu, C. Wang, and H. Zeng, "Research on the comparison of Flood DDoS and Low-rate DDoS," *2011 International Conference on Multimedia Technology*," Hangzhou, China. IEEE, 2011. doi: 10.1109/ICMT.2011.6002141 pp. 5503–5506. 12

[49] Wired, "New Breed of Attack Zombies Lurk. Accessed at: 15 Mar. 2022," 2001. [Online]. Available: https://www.wired.com/2001/05/new-breed-of-attack-zombies-lurk/ 12

[50] A. P. Haripriya and K. Kulothungan, "Secure-MQTT: an efficient fuzzy logic-based approach to detect DoS attack in MQTT protocol for internet of things," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–15, 2019. doi: 10.1186/s13638-019-1402-8 14, 90

[51] S. Iranmanesh, F. S. Abkenar, A. Jamalipour, and R. Raad, "A Heuristic Distributed Scheme to Detect Falsification of Mobility Patterns in Internet of Vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 719–727, 2022. doi: 10.1109/JIOT.2021.3085315 14

[52] S. Ramesh, C. Yaashuwanth, K. Prathibanandhi, A. R. Basha, and T. Jayasankar, "An optimized deep neural network based DoS attack detection in wireless video sensor network," *Journal of Ambient Intelligence and Humanized Computing*, 2021. doi: 10.1007/s12652-020-02763-9 14

[53] R. SaiSindhuTheja and G. K. Shyam, "An efficient metaheuristic algorithm based feature selection and recurrent neural network for DoS attack detection in cloud computing environment," *Applied Soft Computing*, vol. 100, p. 106997, 2021. doi: 10.1016/j.asoc.2020.106997 14

[54] F. Hussain, S. G. Abbas, G. A. Shah, I. M. Pires, U. U. Fayyaz, F. Shahzad, N. M. Garcia, and E. Zdravevski, "A Framework for Malicious Traffic Detection in IoT Healthcare Environment," *Sensors*, vol. 21, no. 9, 2021. doi: 10.3390/s21093025 14

[55] Y. Chen and K. Hwang, "Collaborative detection and filtering of shrew DDoS attacks using spectral analysis," *Journal of Parallel and Distributed Computing*, vol. 66, no. 9, pp. 1137–1151, 2006. doi: 10.1016/j.jpdc.2006.04.007 17

[56] H. Sun, J. Lui, and D. K. Y. Yau, "Distributed mechanism in detecting and defending against the low-rate TCP attack," *Computer Networks*, vol. 50, no. 13, pp. 2312–2330, 2006. doi: 10.1016/j.comnet.2005.09.016 17

[57] Z. Wu, J. Lei, D. Yao, M. Wang, and S. M. Musa, "Chaos-based detection of LDoS attacks," *Journal of Systems and Software*, vol. 86, no. 1, pp. 211–221, 2013. doi: 10.1016/j.jss.2012.07.065 17

[58] S. S. Kanhere and A. Naveed, "A novel tuneable low-intensity adversarial attack," *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)l*," Sydney, NSW, Australia. IEEE, 2005. doi: 10.1109/LCN.2005.14 pp. 794–801. 17, 23

[59] X. Luo and R. K. C. Chang, "On a New Class of Pulsing Denial-of-Service Attacks and the Defense. Accessed at: 07 out. 2018," 2005. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2017/09/On-a-New-Class-of-Pulsing-Denial-of-Service-Attacks-and-the-Defense-Xiapu-Luo.pdf 17, 23, 41, 42

[60] J. Luo and X. Yang, "The NewShrew attack: A new type of low-rate TCP-Targeted DoS attack," *2014 IEEE International Conference on Communications (ICC)*," Sydney, NSW, Australia. IEEE, 2014. doi: 10.1109/ICC.2014.6883403 pp. 713–718. 18, 23, 49

[61] M. Guirguis, A. Bestavros, and I. Matta, "On the impact of low-rate attacks," *2006 IEEE International Conference on Communications*," Istanbul, Turkey, vol. 5. IEEE, 2006. doi: 10.1109/ICC.2006.255115 pp. 2316–2321. 18, 23

[62] M. Yue, Z. Wu, and M. Wang, "A new exploration of FB-shrew attack," *Communications Letters*, vol. 20, no. 10, pp. 1987–1990, 2016. doi: 10.1109/LCOMM.2016.2596278 18

[63] M. Yue, M. Wang, and Z. Wu, "Low-high burst: a double potency varying-rtt based full-buffer shrew attack model," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, pp. 2285–2300, 2019. doi: 10.1109/TDSC.2019.2948167 18

[64] S. A. Arunmozhi and Y. Venkataramani, "A Flow Monitoring Scheme to Defend Reduction-of-Quality (RoQ) Attacks in Mobile Ad-hoc Networks," *Information Security Journal: A Global Perspective*, vol. 19, no. 5, pp. 263–272, 2010. doi: 10.1080/19393555.2010.514651 18, 34, 37, 52, 54

[65] M. Guirguis, A. Bestavros, I. Matta, and Y. Zhang, "Adversarial exploits of end-systems adaptation dynamics," *Journal of Parallel and Distributed Computing*, vol. 67, no. 3, pp. 318–335, 2007. doi: 10.1016/j.jpdc.2006.10.005 18, 35, 37, 52, 53

[66] Y. Ke, C. Chen, H. Hsiao, A. Perrig, and V. Sekar, "Cicadas: Congesting the internet with coordinated and decentralized pulsating attacks," *11th ACM on Asia Conference on Computer and Communications Security*," Xi'an, China, 2016, pp. 699–710. 19, 23

[67] S. Shekyan, "slowhttptest. Accessed at: 14 Abr. 2022," 2011. [Online]. Available: https://github.com/shekyan/slowhttptest/wiki 19, 45

[68] M. M. Najafabadi, T. M. Khoshgoftaar, A. Napolitano, and C. Wheelus, "RUDY Attack: Detection at the Network Level and Its Important Features," *Twenty-Ninth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2016)*," Florida, USA, 2016, pp. 288–293. 19, 23, 41, 42

[69] Cloudfare, "R U Dead Yet (Já morreu)? Ataque (R.U.D.Y.). Accessed at: 14 out. 2021," 2018. [Online]. Available: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/r-u-dead-yet-rudy/ 19

[70] S. Shekyan, "Are You Ready For Slow Reading?. Accessed at: 14 out. 2022," 2012. [Online]. Available: https://blog.shekyan.com/2012/01/are-you-ready-for-slow-reading.html 20, 23

[71] J. Park, K. Iwai, H. Tanaka, and T. Kurokawa, "Analysis of slow read dos attack and countermeasures on web servers," *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, vol. 4, no. 2, pp. 339–353, 2015. 20

[72] C. Kemp, C. Calvert, and T. Khoshgoftaar, "Utilizing netflow data to detect slow read attacks," *2018 IEEE International Conference on Information Reuse and Integration (IRI)*," Salt Lake City, UT, USA. IEEE, 2018. doi: 10.1109/IRI.2018.00023 pp. 108–116. 20

[73] E. Cambiaso, G. Papaleo, G. Chiola, and M. Aiello, "Designing and modeling the slow next DoS attack," *Computational Intelligence in Security for Information Systems Conference*," Burgos, Spain. Springer, 2015. doi: 10.1007/978-3-319-19713-5_22 pp. 249–259. 21, 23, 49

[74] M. Aiello, G. Papaleo, and E. Cambiaso, "SlowReq: a weapon for cyberwarfare operations. Characteristics, limits, performance, remediations," *International Joint Conference SOCO'13-CISIS'13-ICEUTE'13*," Salamanca, Spain. Springer, 2014. doi: 10.1007/978-3-319-01854-6_55 pp. 537–546. 22, 23, 49

[75] E. Cambiaso, G. Papaleo, and M. Aiello, "Slowcomm: Design, development and performance evaluation of a new slow DoS attack," *Journal of Information Security and Applications*, vol. 35, pp. 23–31, 2017. doi: 10.1016/j.jisa.2017.05.005 22, 49

[76] Cambiaso, Enrico and Papaleo, Gianluca and Aiello, Maurizio, "Slowdroid: Turning a smartphone into a mobile attack vector," *2014 International Conference on Future Internet of Things and Cloud*," Barcelona, Spain. IEEE, 2014. doi: 10.1109/FiCloud.2014.72 pp. 405–410. 22, 23, 45

[77] N. Tripathi and N. Hubballi, "Slow rate denial of service attacks against HTTP/2 and detection," *Computers & security*, vol. 52, pp. 255–272, 2018. doi: 10.1016/j.cose.2017.09.009 22, 23, 42, 43, 48, 49

[78] G. Maciá-Fernández, J. E. Díaz-Verdejo, and P. García-Teodoro, "Mathematical Model for Low-Rate DoS Attacks Against Application Servers," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 3, pp. 519–529, 2009. doi: 10.1109/TIFS.2009.2024719 24, 49

[79] G. Yang, M. Gerla, and M. Y. Sanadidi, "Defense against Low-rate TCP-targeted Denial-of-Service Attacks," *ISCC 2004. Ninth International Symposium on Computers And Communications (IEEE Cat. No.04TH8769)*, vol. 1, pp. 345–350, 2004. doi: 10.1109/ISCC.2004.1358428 25, 32

[80] H. Sun, J. C. Lui, and D. K. Yau, "Defending against low-rate TCP attacks: Dynamic detection and protection," *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004*," Berlin, Germany. IEEE, 2004. doi: 10.1109/ICNP.2004.1348110 pp. 196–205. 25, 32

[81] Y. Kwok, R. Tripathi, Y. Chen, and K. Hwang, "HAWK: Halting Anomalies with Weighted Choking to Rescue Well-Behaved TCP Sessions from Shrew DDoS Attacks," *International Conference on Networking and Mobile Computing*, pp. 423–432, 2005. doi: 10.1007/11534310 25, 32

[82] Y. Chen, K. Hwang, and Y. Kwok, "Filtering of Shrew DDoS Attacks in Frequency Domain," *The IEEE Conference on Local Computer Networks 30th Anniversary*

(LCN'05)l," Sydney, NSW, Australia. IEEE, 2005. doi: 10.1109/LCN.2005.70 pp. 1–8. 26, 32

[83] A. Shevtekar, K. Anantharam, and N. Ansari, "Low rate TCP denial-of-service attack detection at edge routers," *IEEE Communications Letters*, vol. 9, no. 4, pp. 363–365, 2005. doi: 10.1109/LCOMM.2005.1413635 26, 32

[84] S. Sarat and A. Terzis, "On the Effect of Router Buffer Sizes on Low-Rate Denial of Service Attacks," *14th International Conference on Computer Communications and Networks, 2005. ICCCN 2005*," San Diego, CA, USA. IEEE, 2005. doi: 10.1109/ICCCN.2005.1523867 pp. 281–286. 26, 32

[85] A. Shevtekar, J. Stille, and N. Ansari, "On the impacts of low rate DoS attacks on VoIP traffic," *Security and Communication Networks*, vol. 1, no. 1, pp. 45–56, 2008. doi: 10.1002/sec.7 26, 32

[86] Z. Liu and L. Guan, "Attack Simulation and Signature Extraction of Low-Rate DoS," *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*," Jinggangshan, China, 2010. doi: 10.1109/IITSI.2010.38 pp. 544–548. 26, 32

[87] C. Zhang, J. Yin, Z. Cai, and W. Chen, "RRED: Robust RED Algorithm to Counter Low-Rate Denial-of-Service Attacks," *IEEE Communications Letters*, vol. 14, no. 5, pp. 489–491, 2010. doi: 10.1109/LCOMM.2010.05.091407 27, 32

[88] H. Hu, J. Zhang, B. Liu, L. Chen, and X. Chen, "Simulation and analysis of distributed low-rate denial-of-service attacks," *5th International Conference on Computer Sciences and Convergence Information Technology*," Seoul. IEEE, 2010. doi: 10.1109/ICCIT.2010.5711129 pp. 620–626. 27, 32

[89] H. Kumawat and G. Meena, "Characterization, Detection and Mitigation of Low-Rate DoS attack," *ICTCS '14: Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies*," Udaipur, Rajasthan, India. ACM, 2014. doi: 10.1145/2677855.2677924 p. 69. 27, 32

[90] J. Singh, S. Gupta, and L. Kaur, "A MAC Layer Based Defense Architecture for Reduction of Quality (RoQ) Attacks in Wireless LAN," *International Journal of Computer Science and Information Security*, vol. 7, no. 1, 2010. doi: 10.48550/arXiv.1002.2423 pp. 284–291. 27, 32

[91] Z. Wu, M. Yue, D. Li, and K. Xie, "SEDP-based detection of low-rate DoS attacks," *International Journal of Communication Systems*, vol. 28, no. 11, pp. 1772–1788, 2015. doi: 10.1002/dac.2783 27, 32

[92] Y. Xiang, K. Li, and W. Zhou, "Low-rate DDoS attacks detection and traceback by using new information metrics," *IEEE transactions on information forensics and*

security, vol. 6, no. 2. IEEE, 2011. doi: 10.1109/TIFS.2011.2107320 pp. 426–437. 27, 30, 32, 64

[93] K. S. Sahoo, D. Puthal, M. Tiwary, J. J. P. C. Rodrigues, B. Sahoo, and R. Dash, "An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics," *Future Generation Computer Systems*, vol. 89, pp. 685–697, 2018. doi: 10.1016/j.future.2018.07.017 27, 33, 55

[94] X. Zhang, Z. Wu, J. Chen, and M. Yue, "An adaptive KPCA approach for detecting LDoS attack," *International Journal of Communication Systems*, vol. 30, no. 4, p. e2993, 2017. doi: 10.1002/dac.2993 28, 33

[95] Z. Liu, X. Yin, and H. J. Lee, "A new network flow grouping method for preventing periodic shrew DDoS attacks in cloud computing," *2016 18th International Conference on Advanced Communication Technology (ICACT)*," PyeongChang, Korea (South). IEEE, 2016. doi: 0.1109/ICACT.2016.7423276 pp. 66–69. 28, 33

[96] J. Lin, C. Zhang, Z. Cai, Q. Liu, and J. Yin, "A TCP-friendly AQM algorithm to mitigate low-rate DDoS attacks," *International Journal of Autonomous and Adaptive Communications Systems*, vol. 9, no. 1-2, pp. 149–163, 2016. doi: 10.1504/IJAACS.2016.075391 28, 33

[97] C. Huang, P. Yi, F. Zou, Y. Yao, W. Wang, and T. Zhu, "CCID: Cross-Correlation identity distinction method for detecting shrew DDoS," *Wireless Communications and Mobile Computing*, vol. 2019, 2019. doi: 10.1155/2019/6705347 28, 33

[98] M. Şimşek and A. Şentürk, "Fast and lightweight detection and filtering method for low-rate TCP targeted distributed denial of service (LDDoS) attacks," *International Journal of Communication Systems*, vol. 31, no. 18, p. e3823, 2018. doi: 10.1002/dac.3823 28, 33

[99] M. Siracusano, S. Shiaeles, and B. Ghita, "Detection of LDDoS Attacks Based on TCP Connection Parameters," *2018 Global Information Infrastructure and Networking Symposium (GIIS)*," Thessaloniki, Greece, 2018. doi: 10.1109/GIIS.2018.8635701 pp. 1–6. 29

[100] P. Cotae and R. Rabie, "On a Game Theoretic Approach to Detect the Low-Rate Denial of Service Attacks," *2018 International Conference on Communications (COMM)*," Bucharest, Romania, 2018. doi: 10.1109/ICComm.2018.8484775 pp. 19–26. 29

[101] Z. Liu, X. Yin, R. Yang, and A. Dong, "Early Detection of LDDoS Attacks in IOT Utilizing Locality Sensitive Incremental TSVM Method," *2021 23rd International Conference on Advanced Communication Technology*," PyeongChang Kwangwoon_Do, Republic of Korea, 2021. doi: 10.23919/ICACT51234.2021.9371008 pp. 194–199. 29

[102] G. Kaur and P. Agrawal, "Detection of LDoS attacks using variant of CUSUM and Shiryaev-Roberts's algorithm," *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*," Waknaghat, India, 2016. doi: 10.1109/PDGC.2016.7913177 pp. 363–369. 29, 33

[103] Y. Zhang, Z. Mao, and J. Wang, "Low-Rate TCP-Targeted DoS Attack Disrupts Internet Routing. Accessed at: 07 out. 2018," 2007. [Online]. Available: https://www.ndss-symposium.org/ndss2007/low-rate-tcp-targeted-dos-attack-disrupts-internet-routing/ 29, 32

[104] G. Thatte, U. Mitra, and J. Heidemann, "Detection of low-rate attacks in computer networks," *IEEE INFOCOM Workshops 2008*," Phoenix, AZ, USA. IEEE, 2008. doi: 10.1109/INFOCOM.2008.4544638 pp. 1–6. 30, 32

[105] X. He, C. Papadopoulos, J. Heidemann, U. Mitra, U. Riaz, and A. Hussain, "Spectral analysis of bottleneck traffic," University of Southern California, Tech. Rep., 2005. [Online]. Available: https://www.cs.colostate.edu/~christos/papers/He05a.pdf 30

[106] R. Mathew and V. Katkar, "Software based low rate dos attack detection mechanism," *International journal of computer applications*, vol. 20, no. 6, pp. 14–18, 2011. 30, 32

[107] Z. Wu, R. Hu, and M. Yue, "Flow-oriented detection of low-rate denial of service attacks," *International Journal of Communication Systems*, vol. 29, no. 1, pp. 130–141, 2014. doi: 10.1002/dac.2805 30, 32

[108] N. Agrawal and S. Tapaswi, "An SDN-Assisted Defense Mechanism for the Shrew DDoS Attack in a Cloud Computing Environment," *Journal of Network and Systems Management*, vol. 29, no. 2, pp. 1–28, 2021. doi: 10.1007/s10922-020-09580-7 30, 33

[109] D. Boro, M. Haloi, and D. K. Bhattacharyya, "A fast self-similarity matrix-based method for shrew DDoS attack detection," *Information Security Journal: A Global Perspective*, vol. 29, no. 2, pp. 73–90, 2020. doi: 10.1080/19393555.2020.1715514 30, 33

[110] D. Tang, Y. Yan, S. Zhang, J. Chen, and Z. Qin, "Performance and features: mitigating the low-rate TCP-targeted DoS attack via SDN," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 428–444, 2021. doi: 10.1109/JSAC.2021.3126053 31, 33

[111] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Information metrics for low-rate DDoS attack detection: A comparative evaluation," *2014 Seventh International Conference on Contemporary Computing (IC3)*," Noida, India, 2014. doi: 10.1109/IC3.2014.6897151 pp. 80–84. 31, 32

[112] K. Bhushan and B. B. Gupta, "Hypothesis test for low-rate DDoS attack detection in cloud computing environment," *Procedia computer science*, vol. 132, pp. 947–955, 2018. doi: 10.1016/j.procs.2018.05.110 31, 33

[113] Z. Wu, L. Liu, and X. Liu, "The approach of detecting LDoS attack based on correlative parameters," *International Conference on Multimedia Technology*," Hangzhou, China. IEEE, 2011. doi: 10.1109/ICMT.2011.6003045 pp. 5587–5590. 31, 32

[114] K. Chen, H. Liu, and X. Chen, "EBDT: A method for detecting LDoS attack," *2012 IEEE International Conference on Information and Automation*," Shenyang, China, 2012. doi: 0.1109/ICInfA.2012.6246912 pp. 911–916. 32, 34

[115] C. Zhang, Z. Cai, W. Chen, X. Luo, and J. Yin, "Flow level detection and filtering of low-rate DDoS," *Computer Networks*, vol. 56, no. 15. Elsevier, 2012. doi: 10.1016/j.comnet.2012.07.003 pp. 3417–3431. 32, 34, 49

[116] Z. Wu, L. Zhang, and M. Yue, "Low-rate DoS attacks detection based on network multifractal," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 5, pp. 559–567, 2016. doi: 10.1109/TDSC.2015.2443807 32, 34

[117] D. Tang, S. Zhang, J. Chen, and X. Wang, "The detection of low-rate DoS attacks using the SADBSCAN algorithm," *Information Sciences*, vol. 565, pp. 229–247, 2021. doi: 10.1016/j.ins.2021.02.038 33, 34

[118] W. Ren, D.-Y. Yeung, H. Jin, and M. Yang, "Pulsing RoQ DDoS attack and defense scheme in mobile ad hoc networks," *International Journal of Network Security*, vol. 4, no. 2, pp. 227–234, 2007. 35, 37, 52, 53

[119] S. Gulati and A. S. Dhaliwal, "Mitigating RoQ Attacks using Flow Monitoring Method," *International Journal of Engineering Trends and Technology*, vol. 4, pp. 4074–4079, 2013. 36, 37, 52, 54

[120] K. Wen, J. Yang, F. Cheng, C. Li, Z. Wang, and H. Yin, "Two-stage detection algorithm for RoQ attack based on localized periodicity analysis of traffic anomaly," *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*," Shanghai, China. IEEE, 2014. doi: 10.1109/ICCCN.2014.6911829 pp. 1–6. 36, 37, 49, 52, 54

[121] G. Yu, T. Li, J. Wei, and C. Liu, "Assessment of Reduction of Quality Attacks on Mobile IP Networks," *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*," Guangzhou, China. IEEE, 2017. doi: 10.1109/ISPA/IUCC.2017.00073 pp. 449–453. 36, 37, 49, 52, 54

[122] H. Chen, M. Liu, and F. Zhongchuan, "Using Improved Hilbert–Huang Transformation Method to Detect Routing-Layer Reduce of Quality Attack in Wireless Sensor Network," *Wireless Personal Communications*, vol. 104, no. 2, pp. 595–615, 2019. doi: 10.1007/s11277-018-6036-3 36, 37, 50, 52, 54

[123] B. Liu, D. Tang, Y. Yan, Z. Zheng, S. Zhang, and J. Zhou, "TS-SVM: Detect LDoS Attack in SDN Based on Two-step Self-adjusting SVM," *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*," Shenyang, China. IEEE, 2021. doi: 10.1109/TrustCom53373.2021.00100 pp. 678–685. 37, 38, 44

[124] M. Aiello, E. Cambiaso, S. Scaglione, and G. Papaleo, "A similarity based approach for application DoS attacks detection," *2013 IEEE Symposium on Computers and Communications (ISCC)*," Split, Croatia. IEEE, 2013. doi: 10.1109/ISCC.2013.6754984 pp. 430–435. 38, 40, 81, 82, 84

[125] Y. G. Dantas, V. Nigam, and I. E. Fonseca, "A selective defense for application layer ddos attacks," *2014 IEEE Joint Intelligence and Security Informatics Conference*," The Hague, Netherlands. IEEE, 2014. doi: 10.1109/JISIC.2014.21 pp. 75–82. 38, 40, 81, 83, 84

[126] M. Mongelli, M. Aiello, E. Cambiaso, and G. Papaleo, "Detection of DoS attacks through Fourier transform and mutual information," *2015 IEEE International Conference on Communications (ICC)*," London, UK. IEEE, 2015. doi: 10.1109/ICC.2015.7249476 pp. 7204–7209. 39, 40, 81, 83, 84

[127] V. Katkar, A. Zinjade, S. Dalvi, T. Bafna, and R. Mahajan, "Detection of DoS/DDoS attack against HTTP servers using naive Bayesian," *2015 International Conference on Computing Communication Control and Automation*," Pune, India. IEEE, 2015. doi: 10.1109/ICCUBEA.2015.60 pp. 280–285. 39, 40, 81, 83, 84

[128] A. Aqil, A. O. Atya, T. Jaeger, S. V. Krishnamurthy, K. Levitt, P. D. McDaniel, J. Rowe, and A. Swami, "Detection of stealthy TCP-based DoS attacks," *MILCOM 2015 - 2015 IEEE Military Communications Conference*," Tampa, FL, USA. IEEE, 2015. doi: 10.1109/MILCOM.2015.7357467 pp. 348–353. 39, 40, 81, 83, 84

[129] K. J. Singh and T. De, "MLP-GA based algorithm to detect application layer DDoS attack," *Journal of Information Security and Applications*, vol. 36, pp. 145–153, 2017. doi: 10.1016/j.jisa.2017.09.004 39, 40, 48, 51, 55, 81, 83, 84

[130] V. d. S. Faria, J. A. Gonçalves, C. A. M. d. Silva, G. d. B. Vieira, and D. M. Mascarenhas, "SDToW: A Slowloris Detecting Tool for WMNs," *Information*, vol. 11, no. 12, p. 544, 2020. doi: 10.3390/info11120544 39, 40, 45, 46, 81, 83, 84

[131] P. Cotae, M. Kang, and A. Velazquez, "Multiple time series fisher periodicity test for the detection of the distributed new shrew attacks," *2016 International Con-*

ference on Communications (COMM)," Bucharest, Romania. IEEE, 2016. doi: 10.1109/ICComm.2016.7528307 pp. 9–14. 41, 42

[132] X. Luo, E. W. Chan, and R. K. Chang, "Detecting pulsing denial-of-service attacks with nondeterministic attack intervals," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, p. 13, 2009. doi: 10.1155/2009/256821 41, 42

[133] R. Xie, M. Xu, J. Cao, and Q. Li, "SoftGuard: Defend Against the Low-Rate TCP Attack in SDN," *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*," Shanghai, China, 2019. doi: 10.1109/ICC.2019.8761806 pp. 1–6. 44

[134] T. V. Phan, T. M. R. Gias, S. T. Islam, T. T. Huong, N. H. Thanh, and T. Bauschert, "Q-MIND: Defeating Stealthy DoS Attacks in SDN with a Machine-Learning Based Defense Framework," *2019 IEEE Global Communications Conference (GLOBECOM)*," Waikoloa, HI, USA, 2019. doi: 10.1109/GLOBECOM38437.2019.9013585 pp. 1–6. 44

[135] J. A. Pérez-Díaz, I. A. Valdovinos, K.-K. R. Choo, and D. Zhu, "A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning," *IEEE Access*, vol. 8, pp. 155 859–155 872, 2020. doi: 10.1109/ACCESS.2020.3019330 44

[136] J. F. Balarezo, S. Wang, K. G. Chavez, A. Al-Hourani, J. Fu, and K. Sithamparanathan, "Low-rate TCP DDoS Attack Model in the Southbound Channel of Software Defined Networks," *2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS)*," Adelaide, SA, Australia, 2020. doi: 10.1109/ICSPCS50536.2020.9310040 pp. 1–10. 44

[137] D. Tang, S. Zhang, Y. Yan, J. Chen, and Z. Qin, "Real-time Detection and Mitigation of LDoS Attacks in the SDN Using the HGB-FP Algorithm," *IEEE Transactions on Services Computing*, pp. 3471–3484, 2021. doi: 10.1109/TSC.2021.3102046 44

[138] H. S. Ilango, M. Ma, and R. Su, "Low Rate DoS Attack Detection in IoT-SDN using Deep Learning," *2021 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing Communications (GreenCom) and IEEE Cyber, Physical Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*," Melbourne, Australia, 2021. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics53846.2021.00031 pp. 115–120. 44

[139] X. Li, N. Luo, D. Tang, Z. Zheng, Z. Qin, and X. Gao, "BA-BNN: Detect LDoS Attacks in SDN Based on Bat Algorithm and BP Neural Network," *2021 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom)*," New York City, NY, USA, 2021. doi:

10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00050 pp. 300–307. 44

[140] O. Andersson, “Original-Slowloris-HTTP-DoS. Accessed at: 14 out. 2022,” 2018. [Online]. Available: https://github.com/Ogglas/Orignal-Slowloris-HTTP-DoS 45

[141] C. Gilbert, “PyLoris: A testing tool for web server DoS vulnerabilites. Accessed at: 14 out. 2022,” 2008. [Online]. Available: https://web.archive.org/web/20090715100428/http://motomastyle.com/pyloris/ 45

[142] cyberwar4iran, “How to help take down gerdab.ir in 5 easy steps. Accessed at: 14 out. 2022,” 2009. [Online]. Available: http://cyberwar4iran.blogspot.com/ 45

[143] S. T. Institute, “Apache HTTP DoS tool released. Accessed at: 13 Fev. 2022,” 2009. [Online]. Available: https://isc.sans.edu/diary/Apache+HTTP+DoS+tool+released/6601 45

[144] G. Lyon, “apache and squid dos. Accessed at: 13 Fev. 2022,” 2009. [Online]. Available: https://seclists.org/fulldisclosure/2009/Jun/207 45

[145] J. Leitch, “HTTP Bog: A slow HTTP denial-of-service tool. Accessed at: 14 out. 2022,” 2018. [Online]. Available: https://www.softpedia.com/get/Internet/Servers/Server-Tools/HTTP-Bog.shtml 45

[146] Torshammer, “Tor’s Hammer - Slow POST Denial Of Service Testing Tool. Accessed at: 14 out. 2022,” 2012. [Online]. Available: https://sourceforge.net/projects/torshammer/ 45

[147] A. Valialkin, “Goloris - slowloris for nginx DoS. Accessed at: 13 Fev. 2022,” 2014. [Online]. Available: https://github.com/valyala/goloris 45

[148] G. Yaltirakli, “slowloris.py - Simple slowloris in Python. Accessed at: 13 Fev. 2022,” 2015. [Online]. Available: https://github.com/gkbrk/slowloris 45

[149] Wanessa, “A influência do mobile e o papel do e-commerce no crescimento econômico. Accessed at: 19 Abr. 2021,” 2019. [Online]. Available: https://www.jornalcontabil.com.br/noticia/a-influencia-do-mobile-e-o-papel-do-e-commerce-no-crescimento-economico/ 45

[150] A. Menon, “Cyphon-DoS. Accessed at: 15 Nov. 2021,” 2018. [Online]. Available: https://github.com/abila5h/Cyphon-DoS 45

[151] Sloww, “Sloww. Accessed at: 13 Fev. 2021,” 2018. [Online]. Available: https://github.com/ethanent/sloww 45

[152] B. Shmali, “Dotloris. Accessed at: 13 Fev. 2021,” 2018. [Online]. Available: https://github.com/bass3l/dotloris 45

[153] H. Hacker, “Pwnloris: An improved slowloris DoS tool. Accessed at: 14 out. 2022,” 2018. [Online]. Available: https://github.com/h0ussni/pwnloris 45

[154] W. Park and S. Ahn, “Performance comparison and detection analysis in snort and suricata environment,” *Wireless Personal Communications*, vol. 94, no. 2, pp. 241–252, 2017. doi: 10.1007/s11277-016-3209-9 45

[155] T. E. de Sousa Araújo, F. M. Matos, and J. A. Moreira, “Intrusion detection systems’ performance for distributed denial-of-service attack,” *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*,” Pucon, Chile. IEEE, 2017. doi: 10.1109/CHILECON.2017.8229519 pp. 1–6. 45, 46, 83, 84

[156] R. K. Sharma, B. Issac, and H. K. Kalita, “Intrusion detection and response system inspired by the defense mechanism of plants,” *IEEE Access*, vol. 7, pp. 52 427–52 439, 2019. doi: 10.1109/ACCESS.2019.2912114 46

[157] C. Townsley, “Are businesses getting complacent when it comes to DDoS mitigation?” *Network Security*, vol. 2018, no. 6, pp. 6–9, 2018. doi: 10.1016/S1353-4858(18)30054-0 48

[158] A. Chadd, “DDoS attacks: past, present and future,” *Network Security*, vol. 2018, no. 7, pp. 13–15, 2018. doi: 10.1016/S1353-4858(18)30069-2 48

[159] S. Newman, “Under the radar: the danger of stealthy DDoS attacks,” *Network Security*, vol. 2019, no. 2, pp. 18–19, 2019. doi: 10.1016/S1353-4858(19)30025-X 48

[160] Akamai, “State of the Internet Reports. Accessed at: 27 Ago. 2021,” 2019. [Online]. Available: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/state-of-the-internet-security-ddos-and-application-attacks-2019.pdf 48

[161] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and M. Rajarajan, “Scale inside-out: Rapid mitigation of cloud ddos attacks,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 959–973, 2017. doi: 10.1109/TDSC.2017.2763160 48

[162] A. Wang, W. Chang, S. Chen, and A. Mohaisen, “A data-driven study of DDoS attacks and their dynamics,” *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 8, pp. 648–661, 2015. doi: 10.1109/TDSC.2018.2808344 48

[163] A. Saied, R. E. Overill, and T. Radzik, “Detection of known and unknown DDoS attacks using Artificial Neural Networks,” *Neurocomputing*, vol. 172, pp. 385–393, 2016. doi: 10.1016/j.neucom.2015.04.101 48

[164] B. Harris and R. Hunt, “TCP/IP security threats and attack methods,” *Computer communications*, vol. 22, no. 10, pp. 885–897, 1999. doi: 10.1016/S0140-3664(99)00064-X 48

[165] X. Ma, B. An, M. Zhao, X. Luo, L. Xue, Z. Li, T. Miu, and X. Guan, "Randomized Security Patrolling for Link Flooding Attack Detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, pp. 795–812, 2019. doi: 10.1109/TDSC.2019.2892370 48

[166] H. D'Cruze, P. Wang, R. O. Sbeit, and A. Ray, "A software-defined networking (SDN) approach to mitigating DDoS attacks," vol. 558, pp. 141–145, 2018. doi: 10.1007/978-3-319-54978-1_19 48

[167] O. Osanaiye, K.-K. R. Choo, and M. Dlodlo, "Distributed denial of service (DDoS) resilience in cloud: review and conceptual cloud DDoS mitigation framework," *Journal of Network and Computer Applications*, vol. 67, pp. 147–165, 2016. doi: 10.1016/j.jnca.2016.01.001 48

[168] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed, and W. M. Abdullah, "Comprehensive Review of Artificial Intelligence and Statistical Approaches in Distributed Denial of Service Attack and Defense Methods," *IEEE Access*, vol. 7, pp. 51 691–51 713, 2019. doi: 10.1109/ACCESS.2019.2908998 48, 50

[169] S. Behal, K. Kumar, and M. Sachdeva, "Characterizing DDoS attacks and flash events: Review, research gaps and future directions," *Computer Science Review*, vol. 25, pp. 101–114, 2017. doi: 10.1016/j.cosrev.2017.07.003 48

[170] M. Aamir and M. A. Zaidi, "A survey on DDoS attack and defense strategies: from traditional schemes to current techniques," *Interdisciplinary information sciences*, vol. 19, no. 2, pp. 173–200, 2013. doi: 10.4036/iis.2013.173 48

[171] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, "DDoS attacks in cloud computing: Issues, taxonomy, and future directions," *Computer Communications*, vol. 107, pp. 30–48, 2017. doi: 10.1016/j.comcom.2017.03.010 48

[172] N. Agrawal and S. Tapaswi, "Defense schemes for variants of distributed denial-of-service (DDoS) attacks in cloud computing: A survey," *Information Security Journal: A Global Perspective*, vol. 26, no. 2, pp. 61–73, 2017. doi: 10.1080/19393555.2017.1282995 48

[173] N. S. Rao, K. C. Sekharaiah, and A. A. Rao, "A Survey of Distributed Denial-of-Service (DDoS) Defense Techniques in ISP Domains," *Innovations in Computer Science and Engineering*, vol. 32, pp. 221–230, 2019. doi: 10.1007/978-981-10-8201-6_25 48

[174] A. P. Abidoye and I. C. Obagbuwa, "DDoS attacks in WSNs: detection and countermeasures," *IET Wireless Sensor Systems*, vol. 8, no. 2, pp. 52–59, 2017. doi: 10.1049/iet-wss.2017.0029 48

[175] H. H. R. Sherazi, R. Iqbal, F. Ahmad, Z. A. Khan, and M. H. Chaudhary, "DDoS Attack Detection: A Key Enabler for Sustainable Communication in Internet of Vehicles," *Sustainable Computing: Informatics and Systems*, vol. 23, pp. 13–20, 2019. doi: 10.1016/j.suscom.2019.05.002 48

[176] L. Zhou, H. Guo, and G. Deng, "A fog computing based approach to DDoS mitigation in IIoT systems," *Computers & Security*, vol. 85, pp. 51–62, 2019. doi: 10.1016/j.cose.2019.04.017 48

[177] J. Cui, M. Wang, Y. Luo, and H. Zhong, "DDoS detection and defense mechanism based on cognitive-inspired computing in SDN," *Future Generation Computer Systems*, vol. 97, pp. 275–283, 2019. doi: 10.1016/j.future.2019.02.037 48

[178] R. Sahay, G. Blanc, Z. Zhang, and H. Debar, "ArOMA: An SDN based autonomic DDoS mitigation framework," *Computers & Security*, vol. 70, pp. 482–499, 2017. doi: 10.1016/j.cose.2017.07.008 48

[179] R. Priyadarshini and R. K. Barik, "A deep learning based intelligent framework to mitigate DDoS attack in fog environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, pp. 825–831, 2019. doi: 10.1016/j.jksuci.2019.04.010 48

[180] A. E. Agoni and M. Dlodlo, "IP Spoofing Detection for Preventing DDoS Attack in Fog Computing," *2018 Global Wireless Summit (GWS)*," Chiang Rai, Thailand. IEEE, 2018. doi: 10.1109/GWS.2018.8686626 pp. 43–46. 48

[181] S. Boulevard, "The Largest DDoS Attacks & What You Can Learn From Them. Accessed at: 14 out. 2022," 2019. [Online]. Available: https://securityboulevard.com/2019/08/the-largest-ddos-attacks-what-you-can-learn-from-them/ 48

[182] N. Woolf, "DDoS attack that disrupted internet was largest of its kind in history, experts say. Accessed at: 14 out. 2022," 2016. [Online]. Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet 48

[183] Tripwire, "5 Notable DDoS Attacks of 2017. Accessed at: 11 out. 2022," 2017. [Online]. Available: https://www.tripwire.com/state-of-security/featured/5-notable-ddos-attacks-2017/ 48

[184] D. René, "Top 5 DDoS attacks of all times. Accessed at: 14 out. 2022," 2019. [Online]. Available: https://baltimorepostexaminer.com/top-5-ddos-attacks-of-all-times/2019/08/19 48

[185] A. Adamson, "Wikipedia Site Paralyzed In Several Countries Due To Massive DDOS Attack. Accessed at: 01 Jul. 2022," 2019. [Online]. Available: https://www.techtimes.com/articles/245274/20190908/wikipedia-site-paralyzed-in-several-countries-due-to-massive-ddos-attack.htm 48

[186] ddosattacks.net, "Major DDoS attacks increased 967% this year. Accessed at: 01 Jul. 2022," 2018. [Online]. Available: https://ddosattacks.net/major-ddos-attacks-increased-967-this-year/ 49

[187] I. Magazine, "DDoS Attacks Jump 18% YoY in Q2. Accessed at: 30 Jul. 2022," 2018. [Online]. Available: https://www.infosecurity-magazine.com/news/ddos-attacks-jump-18-yoy-in-q2/ 49

[188] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Power spectrum entropy based detection and mitigation of low-rate dos attacks," *Computer Networks*, vol. 136, pp. 80–94, 2018. doi: 10.1016/j.comnet.2018.02.029 49, 55, 85

[189] E. Damon, J. Dale, E. Laron, J. Mache, N. Land, and R. Weiss, "Hands-on denial of service lab exercises using SlowLoris and RUDY," *InfoSecCD '12: Proceedings of the 2012 Information Security Curriculum Development Conference*," Kennesaw, Georgia. ACM, 2012. doi: 10.1145/2390317.2390321 pp. 21–29. 49

[190] A. Sangodoyin, B. Modu, I. Awan, and J. P. Disso, "An approach to detecting distributed denial of service attacks in software defined networks," *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*," Barcelona, Spain. IEEE, 2018. doi: 10.1109/FiCloud.2018.00069 pp. 436–443. 49

[191] M. Aiello, E. Cambiaso, M. Mongelli, and G. Papaleo, "An on-line intrusion detection approach to identify low-rate DoS attacks," *2014 International Carnahan Conference on Security Technology (ICCST)*," Rome, Italy. IEEE, 2014. doi: 10.1109/CCST.2014.6987039 pp. 1–6. 49

[192] J. Park, K. Iwai, H. Tanaka, and T. Kurokawa, "Analysis of slow read DoS attack," *2014 International Symposium on Information Theory and its Applications*," Victoria, BC, Canada. IEEE, 2014, pp. 60–64. 49

[193] S. Shafieian, M. Zulkernine, and A. Haque, "CloudZombie: Launching and detecting slow-read distributed denial of service attacks from the cloud," *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*," Liverpool, UK. IEEE, 2015. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.261 pp. 1733–1740. 49

[194] S. Hosseini and M. Azizi, "The hybrid technique for DDoS detection with supervised learning algorithms," *Computer Networks*, vol. 158, pp. 35–45, 2019. doi: 10.1016/j.comnet.2019.04.027 50, 51, 82

[195] N. Meti, D. G. Narayan, and V. P. Baligar, "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks," *2017 International Conference on Advances in Computing, Communications and Infor-*

matics (ICACCI)," Udupi, India. IEEE, 2017. doi: 10.1109/ICACCI.2017.8126031 pp. 1366–1371. 50, 51, 82

[196] X. Liang and T. Znati, "On the performance of intelligent techniques for intensive and stealthy DDos detection," *Computer Networks*, vol. 164, p. 106906, 2019. doi: 10.1016/j.comnet.2019.106906 50

[197] M. Aamir and S. M. A. Zaidi, "Clustering based semi-supervised machine learning for DDoS attack classification," *Journal of King Saud University-Computer and Information Sciences*, vol. 33, pp. 436–446, 2019. doi: 10.1016/j.jksuci.2019.02.003 50, 51

[198] A. R. Wani, Q. P. Rana, U. Saxena, and N. Pandey, "Analysis and Detection of DDoS Attacks on Cloud Computing Environment using Machine Learning Techniques," *2019 Amity International Conference on Artificial Intelligence (AICAI)*," Dubai, United Arab Emirates. IEEE, 2019. doi: 10.1109/AICAI.2019.8701238 pp. 870–875. 50, 51

[199] M. Panda, A. Abraham, and M. R. Patra, "Discriminative multinomial naive bayes for network intrusion detection," *2010 Sixth International Conference on Information Assurance and Security*," Atlanta, GA, USA. IEEE, 2010. doi: 10.1109/ISIAS.2010.5604193 pp. 5–10. 50, 82

[200] S. K. Ajagekar and V. Jadhav, "Study on web DDOS attacks detection using multinomial classifier," *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*," Chennai, India. IEEE, 2016. doi: 10.1109/ICCIC.2016.7919656 pp. 1–5. 50

[201] Ajagekar, Shital K. and Jadhav, Vaishali, "Automated approach for DDoS attacks detection based on naive Bayes multinomial classifier," *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*," Tirunelveli, India. IEEE, 2018. doi: 10.1109/ICOEI.2018.8553848 pp. 1–5. 50

[202] I. Sreeram and V. P. K. Vuppala, "HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm," *Applied computing and informatics*, vol. 15, no. 1, pp. 59–66, 2017. doi: 10.1016/j.aci.2017.10.003 51

[203] S. S. Mohammed, R. Hussain, O. Senko, B. Bimaganbetov, J. Lee, F. Hussain, C. A. Kerrache, E. Barka, and M. Z. A. Bhuiyan, "A New Machine Learning-based Collaborative DDoS Mitigation Mechanism in Software-Defined Network," *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*," Limassol, Cyprus. IEEE, 2018. doi: 10.1109/WiMOB.2018.8589104 pp. 1–8. 51

[204] A. M. Alrehan and F. A. Alhaidari, "Machine Learning Techniques to Detect DDoS Attacks on VANET System: A Survey," *2019 2nd International Conference on Computer Applications Information Security (ICCAIS)*," Riyadh, Saudi Arabia. IEEE, 2019. doi: 10.1109/CAIS.2019.8769454 pp. 1–6. 51

[205] J. Hou, P. Fu, Z. Cao, and A. Xu, "Machine Learning Based DDos Detection Through NetFlow Analysis," *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*," Los Angeles, CA, USA. IEEE, 2018. doi: 10.1109/MILCOM.2018.8599738 pp. 1–6. 51

[206] C. Balarengadurai and S. Saraswathi, "Detection of exhaustion attacks over IEEE 802.15.4 MAC layer using fuzzy logic system," *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*," Kochi, India. IEEE, 2012. doi: 10.1109/ISDA.2012.6416593 pp. 527–532. 51

[207] J. C. C. Rodriguez, A. P. Briones, and J. A. Nolazco, "FLF4DoS. Dynamic DDoS Mitigation based on TTL field using fuzzy logic, Cholula, Mexico." IEEE, 2007. doi: 10.1109/CONIELECOMP.2007.19 pp. 12–12. 51

[208] A. Alsirhani, S. Sampalli, and P. Bodorik, "DDoS Detection System: Using a Set of Classification Algorithms Controlled by Fuzzy Logic System in Apache Spark," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 936–949, 2019. doi: 10.1109/TNSM.2019.2929425 51

[209] M. H. Bhuyan and E. Elmroth, "Multi-scale Low-Rate DDoS Attack Detection Using the Generalized Total Variation Metric," *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*," Orlando, FL, USA. IEEE, 2018. doi: 10.1109/ICMLA.2018.00170 pp. 1040–1047. 51

[210] M. Guirguis, A. Bestavros, I. Matta, and Y. Zhang, "Reduction of quality (RoQ) attacks on dynamic load balancers: Vulnerability assessment and design tradeoffs," *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 857–865, 2007. doi: 10.1109/INFCOM.2007.105 52, 53

[211] W. Chen, Y. Zhang, and Y. Wei, "The feasibility of launching reduction of quality (RoQ) attacks in 802.11 wireless networks," *2008 14th IEEE International Conference on Parallel and Distributed Systems*," Melbourne, VIC, Australia. IEEE, 2008. doi: 10.1109/ICPADS.2008.59 pp. 517–524. 52, 53

[212] A. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *IEEE transactions on pattern analysis and machine intelligence*, vol. 19, no. 2, pp. 153–158, 1997. doi: 10.1109/34.574797 55

[213] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," *Computer Networks*, p. 17, 2020. doi: 10.1016/j.comnet.2020.107247 55

[214] C. Khammassi and S. Krichen, "A NSGA2-LR wrapper approach for feature selection in network intrusion detection," *Computer Networks*, p. 18, 2020. doi: 10.1016/j.comnet.2020.107183 55

[215] M. Wang, Y. Lu, and J. Qin, "A dynamic MLP-based DDoS attack detection method using feature selection and feedback," *Computers & Security*, vol. 88, pp. 1–14, 2020. doi: 10.1016/j.cose.2019.101645 55

[216] S. M. T. Nezhad, M. Nazari, and E. A. Gharavol, "A novel DoS and DDoS attacks detection algorithm using ARIMA time series model and chaotic system in computer networks," *IEEE Communications Letters*, vol. 20, no. 4, pp. 700–703, 2016. doi: 10.1109/LCOMM.2016.2517622 55

[217] K. Kumar, R. C. Joshi, and K. Singh, "A distributed Approach using entropy to detect DDoS attacks in ISP domain," *2007 International Conference on Signal Processing, Communications and Networking*," Chennai, India. IEEE, 2007. doi: 10.1109/ICSCN.2007.350758 pp. 331–337. 55

[218] R. Wang, Z. Jia, and L. Ju, "An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking," *2015 IEEE Trustcom/BigDataSE/ISPA*," Helsinki, Finland, vol. 1. IEEE, 2015. doi: 10.1109/Trustcom.2015.389 pp. 310–317. 55

[219] A. T. Lawniczak, H. Wu, and B. Di Stefano, "Entropy based detection of DDoS attacks in packet switching network models," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*," Berlin, Heidelberg, vol. 5. Springer, 2009. doi: 10.1007/978-3-642-02469-6_57 pp. 1810–1822. 55

[220] S. S. Kolahi, A. A. Alghalbi, A. F. Alotaibi, S. S. Ahmed, and D. Lad, "Performance comparison of defense mechanisms against TCP SYN flood DDoS attack," *2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*," St. Petersburg, Russia. IEEE, 2014. doi: 10.1109/ICUMT.2014.7002093 pp. 143–147. 55

[221] J. V. P. Gomes, "Classification of Peer-to-Peer traffic by exploring the heterogeneity of traffic features through entropy," Ph.D. dissertation, p. 208, Universidade da Beira Interior, 2012. 55

[222] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x 55, 56

[223] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning

in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. 57, 87

[224] *Genetic algorithms and fuzzy logic systems: Soft computing perspectives*, Book. World Scientific, 1997, vol. 7. 57

[225] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International journal of human-computer studies*, vol. 51, no. 2, pp. 135–147, 1999. doi: 10.1016/S0020-7373(75)80002-2 59, 89

[226] CAIDA, "Center for Applied Internet Data Analysis. Accessed at: 14 out. 2022," 2016. [Online]. Available: https://data.caida.org/datasets/security/ddos-20070804/$ddostrace.20070804_144936.pcap.gz$ 64

[227] Kathará, "Simple, lightweight, and fast. Accessed at: 14 out. 2022," 2016. [Online]. Available: https://www.kathara.org/index.html 64, 90

[228] R. Iakobashvili and M. Moser, "curl-loader. Accessed at: 14 out. 2022," 2007. [Online]. Available: http://curl-loader.sourceforge.net 64

[229] TShark, "tshark(1) Manual Page. Accessed at: 14 out. 2017," 2018. [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html 64

[230] V. de Miranda Rios, "Manipulated-RoQ attack software. Accessed at: 14 out. 2020," 2020. [Online]. Available: https://github.com/ducarios/M-RoQ 67

[231] T. T. T. Nguyen and G. J. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 1-4, pp. 56–76, 2008. doi: 10.1109/SURV.2008.080406 68

[232] C. L. Calvert and T. M. Khoshgoftaar, "Impact of class distribution on the detection of slow HTTP DoS attacks using Big Data," *Journal of Big Data*, vol. 6, no. 1, p. 67, 2019. doi: 10.1186/s40537-019-0230-3 81

[233] Y. Wu, H. Tseng, W. Yang, and R. Jan, "DDoS detection and traceback with decision tree and grey relational analysis," *International Journal of Ad-Hoc and Ubiquitous Computing*, vol. 7, no. 2, pp. 121–136, 2011. doi: 10.1504/IJAHUC.2011.038998 82

[234] M. Idhammad, K. Afdel, and M. Belouch, "Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest," *Security and Communication Networks*, vol. 2018, 2018. doi: 10.1155/2018/1263123 82

[235] H. A. Alamri and V. Thayananthan, "Bandwidth control mechanism and extreme gradient boosting algorithm for protecting software-defined networks against DDoS attacks," *IEEE Access*, vol. 8, pp. 194 269–194 288, 2020. doi: 10.1109/ACCESS.2020.3033942 82

[236] M. Marvi, A. Arfeen, and R. Uddin, "A generalized machine learning-based model for the detection of DDoS attacks," *International Journal of Network Management*, vol. 31, no. 6, p. e2152, 2021. doi: 10.1002/nem.2152 82

[237] Canadian Institute for Cybersecurity, "Intrusion Detection Evaluation Dataset (CIC-IDS2017)," 2017. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html 82

[238] V. M. Deolindo, B. L. Dalmazo, M. V. B. da Silva, L. R. B. de Oliveira, A. d. B. Silva, L. Z. Granville, L. P. Gaspary, and J. C. Nobre, "Using Quadratic Discriminant Analysis by Intrusion Detection Systems for Port Scan and Slowloris Attack Classification," *Computational Science and Its Applications – ICCSA*, vol. 12951. Springer, 2021. doi: 10.1007/978-3-030-86970-0_14 pp. 188–200. 83, 84

[239] P. Oktivasari, A. R. Zain, M. Agustin, A. Kurniawan, F. arbi Murad, and M. fabian Anshor, "Analysis of Effectiveness of Iptables on Web Server from Slowloris Attack, Jakarta, Indonesia." IEEE, 2022. doi: 10.1109/IC2IE56416.2022.9970143 pp. 215–219. 84, 85

[240] A. A. Murthy, P. M. John, and R. M. B. Kasturi Nagappasetty, "Hypertext transfer protocol performance analysis in traditional and software defined networks during Slowloris attack." *International Journal of Electrical & Computer Engineering*, vol. 13, no. 4, 2023. 84, 85

[241] C. Asch, G. Gálvez, E. Ríos, J. J. Vargas, L. Quesada, G. Barrantes, and A. Lara, "Asynchronous Detection of Slowloris Attacks Via Random Forests," *2021 IEEE V Jornadas Costarricenses de Investigación en Computación e Informática (JoCICI)*," San José, Costa Rica. IEEE, 2021. doi: 10.1109/JoCICI54528.2021.9794346 pp. 1–6. 83

[242] R. K. Batchu and H. Seetha, "A generalized machine learning model for DDoS attacks detection using hybrid feature selection and hyperparameter tuning," *Computer Networks*, vol. 200, p. 108498, 2021. doi: 10.1016/j.comnet.2021.108498 87

[243] O. Sanchez, M. Repetto, A. Carrega, and R. Bolla, "Evaluating ML-based DDoS detection with grid search hyperparameter optimization," *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*," Tokyo, Japan. IEEE, 2021. doi: 10.1109/NetSoft51509.2021.9492633 pp. 402–408. 87

[244] R. Wazirali, "An improved intrusion detection system based on KNN Hyperparameter tuning and cross-validation," *Arabian Journal for Science and Engineering*, vol. 45, no. 12, pp. 10 859–10 873, 2020. doi: 10.1007/s13369-020-04907-7 87

[245] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: the CICIDS2017 case study," *2021 IEEE Security and Pri-*

vacy Workshops (SPW)," San Francisco, CA, USA. IEEE, 2021. doi: 10.1109/SPW53761.2021.00009 pp. 7–12. 87

[246] X. Wei, L. Zhang, H.-Q. Yang, L. Zhang, and Y.-P. Yao, "Machine learning for pore-water pressure time-series prediction: Application of recurrent neural networks," *Geoscience Frontiers*, vol. 12, pp. 453–467, 2021. doi: 10.1016/j.gsf.2020.04.011 87

[247] S. V. J. Rani, I. Ioannou, P. Nagaradjane, C. Christophorou, V. Vassiliou, S. Charan, S. Prakash, N. Parekh, and A. Pitsillides, "Detection of DDoS attacks in D2D communications using machine learning approach," *Computer Communications*, vol. 198, pp. 32–51, 2023. doi: 10.1016/j.comcom.2022.11.013 87

[248] V. de Miranda Rios, P. R. M. Inácio, D. Magoni, and M. M. Freire, "Detection of Slowloris Attacks using Machine Learning Algorithms - dataset. Accessed at: 04 Set. 2022," 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8316038 91