Aprendizado de Máquina Aplicado na Classificação de Alertas de Ataques de DoS em Sistemas de Detecção de Intrusão

Henrique C. F. da Silva[1], Luca B. Pietro[1], Luis G. B. Dario[1], Eduardo A. de Moraes[1], Paulo R. G. Hernandes Junior[1], Emerson R. A. Barea[2]

[1]Faculdade de Tecnologia de Ourinhos (FATEC) – São Paulo / Brasil

[2]Instituto Federal do Tocantins (IFTO) – Tocantins / Brasil

Resumo. *Este trabalho contribui com a evolução dos sistemas de detecção de intrusão ao propor um modelo de aprendizado de máquina capaz de identificar ataques de negação de serviço. Para isso, foi utilizado o algoritmo Random Forest sobre um dataset com tipos variados de registros de ataque, validando a abrangência da proposta. Durante o desenvolvimento do modelo final, foram observadas técnicas para reduzir o número de falsos positivos e negativos, consequentemente atingindo estatísticas de desempenho relevantes. Os resultados preliminares indicaram boa capacidade de reconhecimento dos ataques.*

Abstract. *This work contributes with the evolution of intrusion detection systems by proposing a machine learning model capable of identifying denial of service attacks. For this, the Random Forest algorithm was used on a dataset with varied types of attack records, validating the scope of the proposal. During the development of the final model, we observed techniques to reduce the number of false positives and negatives, consequently reaching relevant performance statistics. Preliminary results indicated a good ability to recognize attacks.*

1. Introdução

A evolução dos sistemas computacionais e sua utilização nas mais diversas áreas vêm contribuindo com o aumento acelerado do tráfego de rede global. Nessa linha, um estudo da Cisco Systems estima que o tráfego atinja 4,8 ZB de dados em 2022, valor 11 vezes superior ao observado em 2012 (437 EB) [Cisco 2018].

Nesse cenário, os ataques que resultam na indisponibilidade dos dados e serviços são ainda mais impactantes, pois representam grande prejuízo às vítimas [Ponemon Institute 2015]. Um exemplo bem conhecido desse tipo de ameaça é o ataque de negação de serviço (*Denial of Service* - DoS), que registrou um aumento considerável na duração e número de eventos ocorridos no território nacional, elevando o Brasil para a quarta posição na lista de países mais afetados pela forma distribuída (*Distributed* DoS - DDoS) desse tipo de ataque, no quarto trimestre de 2018 [Kaspersky 2019].

Historicamente, uma ferramenta que tem se mostrado eficiente na mitigação dos DDoS são os sistemas de detecção de intrusão (*Intrusion Detection System* - IDS) [Rozemblum 2001]. Originalmente, esses sistemas baseavam-se na validação dos fluxos de dados de uma rede, ou *host*, com assinaturas preexistentes que representam alertas de segurança, reportando os possíveis ataques ou até mesmo bloqueando-os em tempo de execução. Com o passar do tempo, os métodos para identificação dos ataques e ações correspondentes evoluíram para adequar-se à realidade das redes [Prajapati et al. 2014] [Shah and Issac 2018],

porém, o desafio de lidar com os falsos positivos - quando o IDS identifica um ataque inexistente, e falsos negativos - quando o IDS não identifica um ataque real, permanecem.

Em paralelo à evolução dos IDS, um ramo da Inteligência Artificial (IA), denominado aprendizado de máquina, evoluiu consideravelmente nos últimos anos. Nele, os sistemas "aprendem" por meio dos dados que lhes são fornecidos, possibilitando que um algoritmo reconheça padrões e escolha a melhor ação a tomar diante de uma situação específica [Nilsson 1996]. Nesse contexto, é pertinente analisar os resultados da utilização do aprendizado de máquina na classificação de ataques DDoS por um IDS, conferindo a viabilidade de sua utilização para esse fim.

Buscando preencher essa lacuna, esse trabalho apresenta um modelo para aprendizado de máquina que visa identificar a ocorrência de ataques DDoS em alertas de IDS. Entre os destaques e aspectos mais relevantes, este trabalho (*i*) utiliza o conjunto de dados de ataques *Intrusion Detection Evaluation Databaset* (CICIDS2017) [Sharafaldin et al. 2018], coletados pelo *Canadian Institute for Cybersecurity* (CIC), como entrada de dados para o algoritmo de aprendizado de máquina; e (*ii*), utiliza o algoritmo *Random Forest*, método de aprendizado conjunto que trabalha sobre a moda de um grupo de árvores de decisão, todas ligeiramente diferentes umas das outras, para construção do modelo.

O restante desse trabalho está organizado da seguinte forma: a Seção 2 trata dos trabalhos relacionados. A Seção 3 apresenta a metodologia utilizada na realização dos experimentos para validação da proposta. A Seção 4 apresenta os resultados, com a discussão das métricas, gráficos e valores obtidos; e a Seção 5 conclui e apresenta sugestões de trabalhos futuros.

2. Trabalhos relacionados

Há uma variedade significativa de trabalhos que exploram o aprendizado de máquina aplicado na classificação de alertas de IDS. Como exemplo, Buczak and Guven 2015 realizaram uma pesquisa literária sobre métodos de aprendizado de máquina e mineração de dados para análise cibernética em apoio à detecção de intrusão. O trabalho apresentou descrições resumidas de cada método, seus usos e complexidade, além de discussões sobre os desafios da área de cibersegurança.

Moraes et al. 2017 apresentaram técnicas de aprendizado supervisionado capazes de reduzir o número de falsos positivos em ataques variados. O trabalho demonstrou a viabilidade do *Random Forest* em lidar com o problema, validando a proposta em uma base de dados real e próprio.

Voltando às pesquisas literárias, Singh and Nagpal 2018 trouxeram um resumo descritivo de artigos que aplicaram técnicas de árvores de decisão e *Random Forest* na classificação de alertas de intrusão em IDS. O trabalho disponibiliza pseudocódigos para o funcionamento de uma *Random Forest* e um *framework* para a implementação do método junto a um IDS.

Por sua vez, Hadi 2018 propôs um modelo que utiliza o algoritmo *Random Forest* para operar *Big Data* gerado por IDS. A base de dados utilizada contém diversos tipos de ataques, incluindo DoS. O trabalho concluiu que o modelo proposto pelo autor atinge melhores resultados do que diferentes modelos previamente propostos, porém, os autores utilizaram a base NSL-KDD, um conjunto de dados menos atual que o utilizado no presente

trabalho.

Por fim, Singh et al. 2019 apresentaram uma metodologia capaz de otimizar o número de falsos alarmes na classificação de alertas de intrusão. O algoritmo proposto foi capaz de obter resultados satisfatórios quando comparado ao algoritmo de árvores de decisão J48. O método proposto utiliza apenas os atributos mais relevantes para classificar os ataques, melhorando o desempenho e diminuindo a quantidade de dados processados. Apesar da acurácia obtida ter sido considerada satisfatória pelos autores, os valores foram superados pelo método proposto nesse trabalho. Além disso, a base de dados utilizada no trabalho foi o KDDCUP99, defasado em relação ao CICIDS2017.

3. Metodologia

Para desenvolvimento da proposta, inicialmente foi realizada uma análise preliminar na base de dados CICIDS2017, em busca dos registros com alertas de ataques do tipo DoS. Nessa fase, observou-se que o arquivo $Wednesday - Working - Hours.csv$ atendia ao requisito.

A Figura 1 apresenta o conteúdo do arquivo correspondente, agrupado por número de ocorrências de cada evento de ataque.

Tipo de *DoS/DDoS*	Número de Registros
Benigno	440031
DoS *GoldenEye*	10293
DoS *Hulk*	231073
DoS *Slowhttptest*	5499
DoS *Slowloris*	5796
Heartbleed	11

Figura 1. Registros do arquivo *Wednesday-Working-Hours.csv* agrupado por tipo de ataque

Devido ao pequeno número de ocorrências da vulnerabilidade *Heartbleed* (CVE-2014-0160), bem como por não se tratar de um ataque de DoS, os registros desse grupo não foram considerados no restante do desenvolvimento desse trabalho. Outro ponto relevante é que, apesar da grande diferença no número de registros dos outros grupos, todos possuem mais de cinco mil ocorrências cada, correspondendo a um número suficiente para que o modelo tenha uma curva de aprendizado satisfatória.

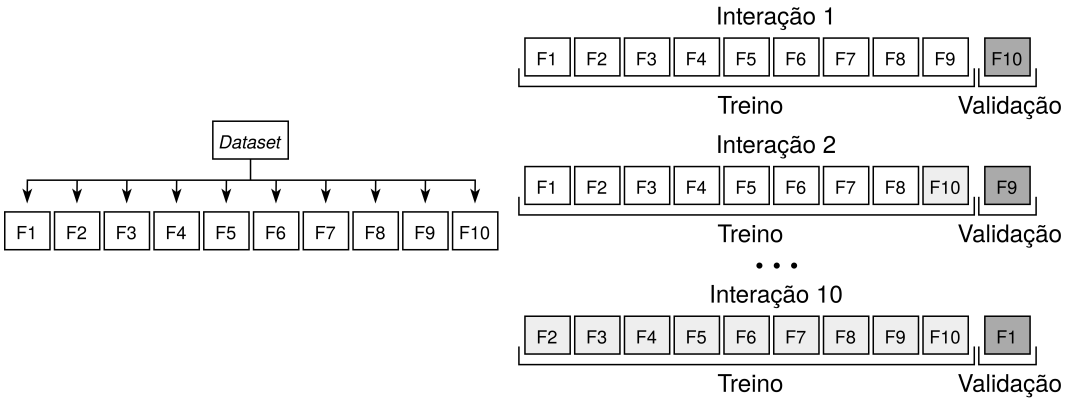
Superada essa fase, o foco passou a ser o planejamento do modelo. Nesse momento foram definidos os métodos de amostragem, os valores dos parâmetros do *Random Forest* e as técnicas para obtenção e plotagem das estatísticas e resultados.

O método de amostragem escolhido foi o *k-folds cross-validation*. Nele, a base é aleatoriamente dividida em uma quantidade k de subgrupos de dimensões equivalentes. Após esse procedimento, um dos subgrupos é reservado para validação posterior, enquanto os outros são utilizados no treinamento do modelo. Esse processo é repetido até que todos subgrupos sejam utilizados na validação do desempenho do modelo [Brownlee 2018].

No método tradicional de amostragem, uma parte dos dados é utilizada apenas para treinamento, enquanto o restante é utilizado somente para validação. A escolha do *k-folds*

cross-validation se deve ao fato da técnica *k-folds* permitir que todos os registros sejam utilizados tanto para treinamento quanto validação.

A Figura 2(a) representa a divisão dos dados em *folds*. Nesse modelo optou-se por utilizar o parâmetro k=10, ou seja, foram gerados 10 subgrupos a partir dos dados originais. A decisão foi baseada no trabalho de Borra and Ciaccio 2010, que compararam a taxa de erro de diversos métodos de amostragem, concluindo que a variação do método *cross-validation* com 10 *folds* apresentou o melhor desempenho.



(a) Diagrama do método *k-folds* (b) Diagrama de funcionamento do *cross-validation*

Figura 2. Diagrama do método *k-folds*

Por fim, a *cross-validation* foi repetida 3 (três) vezes, caracterizando o modelo final como *repeated k-folds cross-validation*. Segundo Kuhn 2014, a repetição da *cross-validation* torna o modelo mais preciso e confiável. A Figura 2(b) ilustra o funcionamento do *cross-validation* durante as diferentes interações de amostragem do modelo.

O número de árvores aleatórias geradas pelo modelo foi estabelecido em 100 (cem). O número de variáveis consideradas na divisão dos nós de uma árvore foi definido como a raiz quadrada do número total de variáveis (colunas), valor padrão da implementação do *Random Forest* no R. Como a base de dados contém 78 colunas de registros, o valor 9 (nove) foi utilizado como parâmetro [Liaw and Wiener 2018].

O modelo foi então executado de acordo com os parâmetros definidos. Para estimar seu desempenho, foram calculadas a Exatidão, *Kappa*, Precisão, Sensibilidade/*Recall*, Especificidade e *F-Score*. A Figura 3 apresenta as fórmulas utilizadas no cálculo das medidas.

Medida	Fórmula
Exatidão	Número de Acertos / Número Total De Tentativas
Kappa	(Exatidão - Exatidão Aleatória) / (1 - Exatidão Aleatória)
Precisão	Verdadeiro Positivo / (Verdadeiro Positivo + Falso Positivo)
Sensibilidade/*Recall*	Verdadeiro Positivo / (Verdadeiro Positivo + Falso Negativo)
Especificidade	Verdadeiro Negativo / (Falso Positivo + Verdadeiro Negativo)
F-Score	2 * ((Precisão * *Recall*) / (Precisão + *Recall*))

Figura 3. Fórmulas das medidas calculadas

Na Figura 3, a Exatidão estima a taxa de registros classificados corretamente

em relação ao número total de registros. O *Kappa* é semelhante a Exatidão, porém considera a chance do registro ter sido classificado corretamente por acaso. A Precisão estima a diferença entre o número de alertas realmente verdadeiros e a quantidade de alertas classificados como verdadeiros. Sensibilidade/*Recall* estimam a taxa de registros realmente verdadeiros que foram classificados corretamente; enquanto a Especificidade estima a taxa de registros realmente falsos que foram classificados corretamente; e o *F-Score* estima a precisão e a cobertura do modelo em uma única métrica.

4. Discussão dos Resultados

Os resultados da validação do modelo proposto incluem a matriz de confusão, métricas estatísticas gerais e individuais por classe, taxas de erro, curvas ROC e valor AUC.

Inicialmente, a Figura 4 apresenta a matriz de confusão dos resultados obtidos, refletindo o comportamento do modelo durante a validação. As linhas correspondem às classificações realizadas pelo algoritmo, enquanto as colunas apresentam os valores reais.

		Valores Reais				
		Benigno	*GoldenEye*	*Hulk*	*Slowhttptest*	*Slowloris*
Predição	Benigno	439719	3	273	34	2
	GoldenEye	12	10247	28	6	0
	Hulk	32	10	231031	0	0
	Slowhttptest	19	3	0	5458	19
	Slowloris	24	0	1	7	5764

Figura 4. Matriz de Confusão dos resultados

Por sua vez, a Figura 5 apresenta as estatísticas gerais do modelo. Nela é possível observar que todos os cálculos resultaram em valores superiores a 99%, com destaque à Exatidão e Especificidade do modelo, que obtiveram resultados superiores a 99,9%. A medida mais complexa dentre as calculadas, o *Kappa*, apresentou valor de 99,87%.

Sensibilidade	**Especificidade**	**Precisão**	***Recall***	***F-Score***
0.997	0.9997	0.9963	0.997	0.9966
	Exatidão		***Kappa***	
	0.9994		0.9987	

Figura 5. Estatísticas gerais do modelo

Conforme os dados da Figura 6(a), nota-se que a Precisão individual por classe ficou entre 99,25% e 99,98%, o menor valor correspondendo aos registros do tipo DoS *Slowhttptest* e o maior ao tipo DoS *Hulk*. Os outros valores variaram entre 99,93% para o Benigno, 99,55% para o DoS *GoldenEye* e 99,45% para o DoS *Slowloris*.

A Figura 6(b) apresenta a Sensibilidade/*Recall* individual por classe. Os valores ficaram entre 99,15% e 99,98%, sendo que o menor valor é referente aos registros do tipo DoS *Slowhttptest* e o maior aos do tipo Benigno. Os demais valores variaram entre 99,84% para o DoS *GoldenEye*, 99,87% para o DoS *Hulk* e 99,64% para o DoS *Slowloris*.

Na Figura 6(c) observamos os valores entre 99,88% e 99,99% para a Especificidade individual por classe. O menor valor foi atingido pela classe Benigno e o maior pela classe DoS *Slowloris*. Os outros valores variaram aproximadamente entre 99,99% para o DoS *GoldenEye*, 99,99% para o DoS *Hulk* e 99,99% para o DoS *Slowhttptest*.

Complementando, a Figura 6(d) apresenta os valores de *F-Score* individual por classe. Os resultados variaram entre 99,2% e 99,95%. O menor valor refere-se ao tipo DoS *Slowhttptest* e o maior para o tipo Benigno. Os outros valores variaram entre 99,70% para o DoS *GoldenEye*, 99,92% para o DoS *Hulk* e 99,54% para o DoS *Slowloris*.

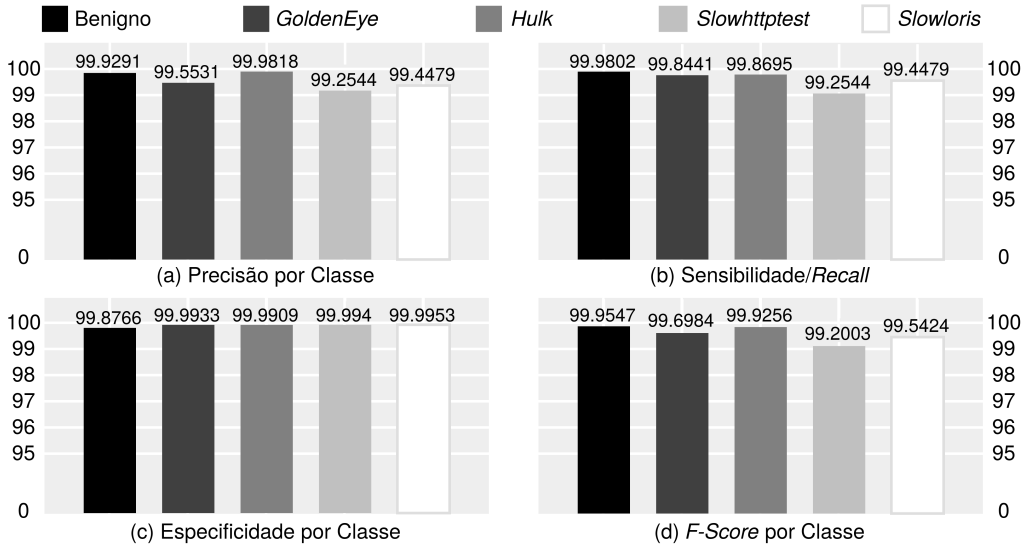


Figura 6. Gráfico de Precisão, Sensibilidade/*Recall*, Especificidade e *F-Score* por Classe

A taxa de erro simples por classe pode ser conferida na Figura 7(a), com valores entre 0,00018 e 0,00745. Nesse caso, quanto mais próximo de 0 (zero) for a taxa de erro, melhor é o desempenho da classe. A média para a taxa de erro simples foi 0,00343, já a média para a taxa de erro *OOB*[1] foi de 0,07%.

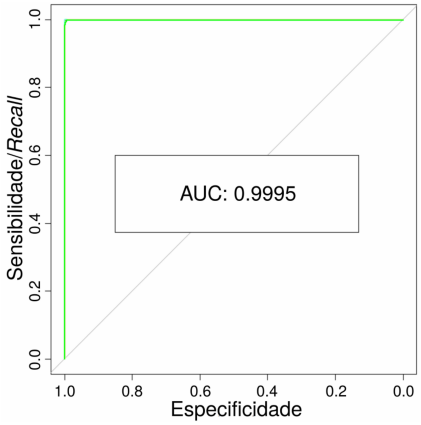
Outra métrica de desempenho importante é a curva ROC (*Receiver Operating Characteristic*) (Figura 7(b)), que representa graficamente a capacidade do modelo em diferir entre duas classes. A curva ROC utiliza como parâmetros as taxas de verdadeiro positivo e falso positivo, junto das métricas Sensibilidade/*Recall* e Especificidade. Também na Figura 7(b), a AUC (*Area Under the ROC Curve*) é o cálculo da área sob uma curva ROC, que pode variar entre 0 e 1. Nesse caso, quanto maior a AUC, melhor é a capacidade do modelo em diferir entre as duas classes. A Figura 7(b) traz a representação das várias curvas ROC do modelo, porém, como cada classe é confrontada com todas as outras, e a execução é repetida três vezes, foram geradas 30 curvas ROC ao todo[2]. A média da AUC para as curvas resultou em 0,9995.

[1]A medida *Out-Of-Bag (OOB)* é um método para avaliar a taxa de erro que considera apenas as predições realizadas sobre registros que o modelo não tenha entrado em contato previamente durante a fase de treino.

[2]Nota-se que as curvas se sobrepuseram quase que completamente, devido à proximidade dos valores.

Classe	Taxa de erro
Benigno	0,0007090410
DoS GoldenEye	0,0044690566
DoS Hulk	0,0001817607
DoS Slowhttptest	0,0074559011
DoS slowloris	0,0055210490
Média	0,003430469
Média *OOB*	0,07%

(a) Taxa de Erro por Classe



(b) Curvas ROC e valor AUC

Figura 7. Curvas ROC e valor AUC

5. Considerações Finais e Trabalhos Futuros

Este trabalho apresentou um modelo de aprendizado de máquina capaz de identificar e classificar ataques de DoS em alertas de IDS. Em seu desenvolvimento foi utilizado o algoritmo *Random Forest* numa parte do *dataset* CICIDS2017 que continha registros de tipos variados de ataques DoS. Ao validar o modelo, observou-se que os resultados foram satisfatórios, pois todas variáveis analisadas apresentaram valores acima de 99% na identificação dos ataques. O maior destaque fica para a exatidão do modelo, que alcançou resultado superior a 99,9%.

Como trabalhos futuros, pretende-se validar o modelo na identificação de outros tipos de ataques existentes nos registros da base de dados CICIDS2017. Outro ponto interessante é o desenvolvimento de um sistema de tomada de decisão automatizado, que execute ações após a classificação de um registro como ameaça.

Referências

Borra, S. and Ciaccio, A. (2010). Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods. *Computational Statistics & Data Analysis*, 54(12):2976–2989.

Brownlee, J. (2018). A Gentle Introduction to k-fold Cross-Validation. Disponível em: `https://machinelearningmastery.com/k-fold-cross-validation/`. Acesso em: 1 nov. 2019.

Buczak, A. L. and Guven, E. (2015). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys And Tutorials*, 18(2):1153–1176.

Cisco (2018). Cisco Visual Networking Index (VNI) - Complete Forecast Update, 2017-2022. Disponível em: `https://www.cisco.com/c/dam/m/en_us/network-intelligence/service-provider/digital-transformation/knowledge-network-webinars/pdfs/1211_BUSINESS_SERVICES_CKN_PDF.pdf`. Acesso em: 19 nov. 2019.

Hadi, A. A. A. (2018). Performance Analysis of Big Data Intrusion Detection System over Random Forest Algorithm. *International Journal of Applied Engineering Research*, 13(2):1520–1527.

Kaspersky (2019). DDoS attacks in Q4 2018. Disponível em: `https://securelist.com/ddos-attacks-in-q4-2018/89565/`. Acesso em: 19 nov. 2019.

Kuhn, M. (2014). Comparing Different Species of Cross-Validation. Disponível em: `http://appliedpredictivemodeling.com/blog/2014/11/27/vpuig01pqbklmi72b8lcl3ij5hj2qm`. Acesso em: 1 nov. 2019.

Liaw, A. and Wiener, M. (2018). Package 'randomForest'. Disponível em: `https://cran.r-project.org/web/packages/randomForest/randomForest.pdf`. Acesso em: 19 nov. 2019.

Moraes, E. A., Tojeiro, C. A. C., Miani, R. S., and Zarpelão, B. B. (2017). Análise de Alertas de Sistemas de Detecção de Intrusão: Uso de Aprendizado Supervisionado na Redução de Alertas Falsos Positivos. *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, 17(1):182–195.

Nilsson, N. J. (1996). *Introduction to Machine Learning: An early draft of a proposed textbook*. Stanford Press, Stanford.

Ponemon Institute (2015). The Cost of Denial-of-Services Attacks. Disponível em: `https://www.akamai.com/us/en/multimedia/documents/content/the-cost-of-denial-of-services-attacks.pdf`. Acesso em: 30 out. 2019.

Prajapati, N. M., Mishra, A., and Bhanodia, P. (2014). Literature survey - ids for ddos attacks. In *2014 Conference on IT in Business, Industry and Government (CSIBIG)*, pages 1–3.

Rozemblum, D. (2001). Understanding Intrusion Detection Systems. Disponível em: `https://www.sans.org/reading-room/whitepapers/detection/understanding-intrusion-detection-systems-337`. Acesso em: 2 nov. 2018.

Shah, S. A. R. and Issac, B. (2018). Performance Comparison of Intrusion Detection Systems and Application of Machine Learning to Snort System. *Future Generation Computer Systems*, 80:157–170.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116.

Singh, K. and Nagpal, B. (2018). Random Forest Algorithm in Intrusion Detection System : A Survey. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 3(5):673–676.

Singh, R. K., Dalal, S., Chauhan, V. K., and Kumar, D. (2019). Optimization of FAR in Intrusion Detection System by using Random Forest Algorithm. In *2nd International Conference on Advanced Computing and Software Engineering*, pages 274–277, Sultanpur.