Toward a Search-Based Approach to Support the Design of Security Tests for Malicious Network Traffic

Davide La Gamba
University of Salerno, Italy
d.lagamba@studenti.unisa.it

Gerardo Iuliano
University of Salerno, Italy
geiuliano@unisa.it

Gilberto Recupito
University of Salerno, Italy
grecupito@unisa.it

Giammaria Giordano
University of Salerno, Italy
giagiordano@unisa.it

Filomena Ferrucci
University of Salerno, Italy
fferrucci@unisa.it

Dario Di Nucci
University of Salerno, Italy
ddinucci@unisa.it

Fabio Palomba
University of Salerno, Italy
fpalomba@unisa.it

ABSTRACT

IoT devices generate and exchange large amounts of data daily, creating significant security and privacy challenges. Security testing, particularly using Machine Learning (ML), helps identify and classify potential malicious network traffic. Previous research has shown how ML can aid in designing security tests for IoT attacks. This ongoing paper introduces a search-based approach using Genetic Algorithms (GAs) to evolve detection rules and detect intrusion attacks. We build on existing GA methods for intrusion detection and compare them with leading ML models. We propose 17 detection rules and demonstrate that while GAs do not fully replace ML, they perform well with ample attack examples and enhance the usability and implementation of deterministic test cases by security testers.

CCS CONCEPTS

• **Software and its engineering** → **Search-based software engineering**; **Software testing and debugging**.

KEYWORDS

Internet-Of-Things; Security Test Code Design; Intrusion Detection Attacks; Genetic Algorithms; Security Testing.

ACM Reference Format:
Davide La Gamba, Gerardo Iuliano, Gilberto Recupito, Giammaria Giordano, Filomena Ferrucci, Dario Di Nucci, and Fabio Palomba. 2024. Toward a Search-Based Approach to Support the Design of Security Tests for Malicious Network Traffic. In *28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), June 18–21, 2024, Salerno, Italy.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3661167.3661288

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
EASE 2024, June 18–21, 2024, Salerno, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1701-7/24/06
https://doi.org/10.1145/3661167.3661288

1 INTRODUCTION

Data generated and exchanged by IoT devices presents significant security challenges, especially in networks [3]. Secure protocols are essential for protecting data integrity and user privacy, while comprehensive testing helps identify and rectify vulnerabilities that attackers could exploit. Due to the extensive adoption of IoT devices, the Software Engineering (SE) community primarily focused on analyzing how *security testing* can help to prevent potential attacks [6] and support practitioners to improve the system's robustness to defend from malicious attacks. Recent research by Giordano et al. [7] analyzed how Artificial Intelligence (AI) is being used to enhance privacy and security in IoT systems. Their study identified a trend in utilizing AI to develop frameworks for security test cases. AI algorithms help security testers create effective test cases, especially through intrusion detection models. These models use specific criteria to detect and classify security threats, discovering vulnerabilities within IoT network security.

Al-fuhaidi et al. [1] employed Genetic Algorithms (GAs) to define detection rules for identifying potentially *DoS* attacks, focusing on the KDDCUP99 dataset [19]. Their GA-based approach targeted DoS attacks and was compared with three established ML methods, yielding similar performance results. While Al-fuhaidi et al. [1] pioneered the application of search-based algorithms, particularly GAs, to define detection rules for identifying potentially malicious traffic, their focus was primarily on DoS attacks using the KDDCUP99 dataset [19] without considering other attacks. This ongoing paper aims to fill the gap by extending Al-fuhaidi et al.[1] providing the following main contributions:

- A set of 17 detection rules for DoS, Probe, U2R, and R2L attacks based on the features of the KDDCUP99 dataset [19];
- A comparison between GA and ML models to detect malicious traffic;
- A public replication package [10] with the data and scripts used in the study available for the research community.

2 INTRUSION DETECTION

Intrusion detection involves identifying unauthorized actions against information systems, known as intrusions [22]. These intrusions can be perpetrated by internal users seeking to elevate their access

privileges or external users attempting to breach the system from outside the network [20]. These activities pose various threats to network security such as:

Denial of Service Attack (DoS) that overwhelms computing or memory resources, rendering them too occupied to handle genuine requests, disrupting regular system functioning, and denying legitimate users access, often employing techniques like Smurf and Ping of Death.

Probing Attack (Probe) aims at collecting data regarding a network of computers, seemingly to bypass its security measures.

User to Root Attack (U2R) describes an exploit in which the attacker begins with a standard user account and then exploits system vulnerabilities to gain root access, escalating privileges beyond their initial level.

Remote to Local Attack (R2L) arises when an attacker, capable of sending packets to a machine over a network but lacking an account on that machine, exploits a vulnerability to attain local access as a user of that machine.

3 RELATED WORK

In the following, we review how AI algorithms can be used to detect intrusion attacks in the KDDCUP99 dataset. Stein et al. [17] utilized GAs for feature selection, significantly enhancing the performance of Decision Tree classifiers by pinpointing the most relevant features. Dong et al. [9] and Li and Wang [11] improved SVM-based models using GAs for feature selection and pre-processing, significantly boosting model performance. Paliwal and Gupta [18] used GAs to create a comprehensive detection rule for all attacks in the dataset, achieving a detection rate of 97%. The most similar research to ours is by Al-fuhaidi et al. [1], who used GAs to develop detection rules specifically for DoS attacks and compared them with three ML techniques—Bayes Network, Decision Tree, and SVM—achieving a high detection rate and low false positive rate.

Building on Al-fuhaidi et al.'s method, we extend the application of GAs to generate detection rules for all attack categories in the KDDCUP99 dataset and broaden the scope by comparing our results to an expanded set of ML algorithms, including Decision Table, Naive Bayes, and Random Forest.

4 RESEARCH METHOD

The *goal* of this study is to analyze to what extent GAs can be used to detect malicious network traffic provided by IoT devices. To achieve this goal, we generated a set of detection rules. The *purpose* is to provide a set of rules that security testers can use to build security tests. The *quality focus* is on GAs and their applicability to detect suspect network traffic in IoT environments and compare it with the performances of existing ML techniques. The *perspective* is for both researchers and practitioners. The former are interested in increasing their knowledge and comprehending the benefits and drawbacks of using GA to discover suspect network traffic. The latter are interested in discovering automatic solutions to implement more deterministic tests. The *context* of our investigation is the *KDDCUP99* [19] dataset, featuring four categories of attacks: Denial of Service Attack (DoS), Probing Attack (Probe), User to Root Attack (U2R), and Remote to Local Attack (R2L), and where the Normal label indicates legitim network traffic.

Based on our motivations and according to our goal, we formulate the following research questions (RQs):

> **RQ$_1$** *To what extent can genetic algorithms generate detection rules to identify DoS attacks?*

With our **RQ**$_1$, we are interested in re-assessing the state-of-the-art by comparing our implementation with the results achieved by Al-fuhaidi et al. [1]. This step is necessary due to the unavailability of a replication package in the original study.

> **RQ$_2$** *What are the performance of GAs in detecting intrusion attacks?*

After assessing the ability of GAs of generating detection rules, we aim to assess the performances of detecting intrusion attacks available in the KDDCUP99 dataset. To address **RQ**$_2$, we compared the performance of GAs with respect to ML algorithms.

We design the study following the guidelines by Wohlin et al. [21] and the *ACM/SIGSOFT Empirical Standards*.[1] Figure 1 illustrates the research process we applied to perform our investigation.

Dataset Description. To achieve our objective, we select the KDDCUP99 dataset. The motivations to select this dataset are fourfold: (i) the dataset is one of the largest publicly available, with over five million connection records obtained by a TCP DUMP data of seven weeks, with over two million of connection records for the test set, (ii) the dataset was analyzed mainly in previous work, resulting in the top-3 of the most used datasets in IoT environments according to the SLR conducted by Giordano et al. [7], (iii) Al-fuhaidi et al. [1] leverages this dataset to perform its analysis, and (iv) KDDCUP99 features five categories of network traffic [19]. Due to the large number of connection records in the dataset, the authors also released a representative sampling composed of 10% of data instances. This sampling dataset is typically used in previous work that investigates similar tasks [1, 15, 16]. We analyzed the dataset characteristics by observing the distribution for each network traffic label (i.e., DoS, Normal, Probe, R2L, and U2R).

An overview of the dataset reveals that out of over 494k connection instances, representing 79% of the instances, are classified as DoS attacks, followed by 20% as Normal traffic. The remaining 1% is distributed among the other three categories (Probe, R2L, and U2R), highlighting the unbalanced nature of the dataset.

Encoding of the Individuals. The individuals in the population are sets of *if (condition) then {action}* rules designed to detect harmful connections, which are formed using the logical AND operator, with numerical features encoded with inequalities, except for categorical ones (e.g., protocolType). Numerical genes are integers bounded by specific minimum and maximum values derived from dataset analysis. Each attack category is encoded separately based on relevant features identified using the *Information Gain* algorithm provided by Kayacık et al. [8]. For instance, for DoS attacks, the chromosome includes 11 parts; some containing single genes corresponding to unique feature ranges and others encoding inequality verses. Feature ranges represent the value bounded by a maximum and a minimum for each feature. The features with equal values are "WrongFragment", "Urgent", "Count" and "SrvCount", and those

[1] Available at: https://github.com/acmsigsoft/EmpiricalStandards

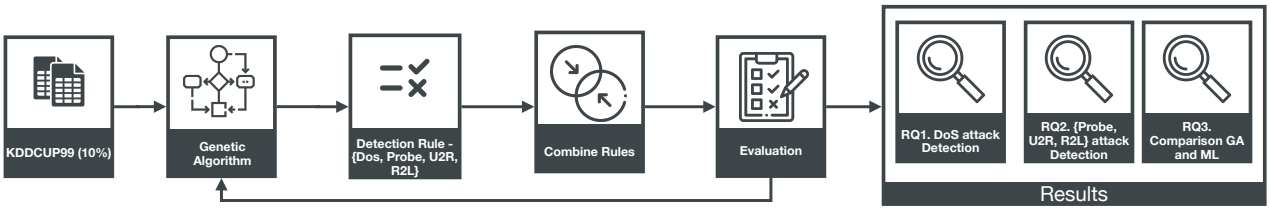

Figure 1: Overview of the research method applied in our study.

expressing percentage values. The last part of the chromosome contains genes related to the inequality verses to be tested. Specifically, "1" represents "≤" while "2" represents "≥".

Fitness Function. To compare our results, we adopt the same fitness function provided by Al-fuhaidi et al. [1].

$$fitness = \frac{detected\_attacks}{total\_attacks} - \frac{false\_attacks}{total\_connections} \quad (1)$$

The variable *detected_attacks* indicates the attacks correctly detected, whereas *false_attacks* indicates the connections misclassified as malicious. *total_attacks* is the number of attacks, and *total_connections* is the total number of network connections, including those malicious. Notably, larger fitness function values indicate rules with a better ability to detect threats in the dataset, so the fitness measure is to be maximized.

Parameters. We used the Java library Jenetics for evolutionary computation. We attempt several parameter configurations, and the final consists of 10% and 90% mutation and crossover probability. The selection operator is the *EliteSelector* [4] to ensure a form of elitism [14] in the population, allowing the best solutions to survive to the next generation without change. The default mutation operator alters individual genes within prefixed ranges to enhance solution exploration and diversity in the population. The population size is set at 1,000, with a stopping criterion set at 1,000 generations. The initial population is generated randomly, defining individuals within a specified range for each gene to ensure diversity.

Validation. We compare the performance of our method with a set of ML algorithms to contextualize and evaluate its performance while replicating Al-fuhaidi et al. [1]. To reinforce the validation step, we decide to include in our experiment the set of ML techniques used by Al-fuhaidi et al.Bayes Network, Decision Tree, and Support Vector Machine and to extend it by including Decision Table, Naive Bayes and Random Forest. We select these techniques due to their popularity, as they have been used in literature for similar tasks on the same dataset [2, 5, 12, 13]. We decided to apply no data balancing technique to maintain the original prevalence ratios of intrusion attacks in the datasets. We ensured consistent observation of all attack types by employing 10-fold cross-validation. This approach allowed the model to robustly evaluate performance across diverse attack scenarios without altering the inherent attack distributions.

5 ANALYSIS OF THE RESULTS

This section describes the results of our work. For the sake of comprehensibility, we split the discussion by research question, including the detection rules and the performance achieved by the experimented approaches.

Table 1: Performance of the GA as # rules increases.

		Performance Metrics						
Rule Type	# Rules	Precision	Recall	F-Measure	Accuracy	Specificity	MCC	FA Rate
DoS	1	1.000	0.207	0.342	0.363	1.000	0.221	0.000
	2	1.000	0.915	0.956	0.932	1.000	0.824	0.000
	3	1.000	0.965	0.982	0.972	1.000	0.918	0.000
	4	1.000	0.965	0.982	0.972	1.000	0.919	0.000
	5	1.000	0.968	0.984	0.974	1.000	0.925	0.000
	6	1.000	0.971	0.985	0.976	1.000	0.931	0.000
	7	1.000	0.972	0.986	0.977	1.000	0.933	0.000
	8	1.000	0.972	0.986	0.978	1.000	0.934	0.000
	9	1.000	0.973	0.986	0.978	1.000	0.935	0.000
	10	1.000	0.978	0.989	0.982	1.000	0.947	0.000
	11	1.000	0.981	0.990	0.985	1.000	0.954	0.000
Probe	12	1.000	0.992	0.996	0.994	**1.000**	0.981	**0.000**
	13	**1.000**	0.994	0.997	0.995	0.999	0.985	0.001
	14	0.999	0.996	0.998	0.996	0.996	0.988	0.004
	15	0.999	0.997	0.998	0.997	0.996	0.990	0.004
U2L	16	0.998	0.998	**0.998**	**0.997**	0.992	**0.990**	0.008
	17	0.994	**0.999**	0.997	0.994	0.974	0.982	0.026

5.1 Detection Rules

We developed 17 detection rules for various network attacks: 11 for DoS attacks, four for Probe attacks, and two for U2R attacks. Additionally, we discovered that combining DoS and Probe attack rules effectively detects R2L attacks.

For certain features such as *ProtocolType*, *Service*, *Flag*, and *Land*, we only used the *equality* operator. This choice was made because the specific nature of these features and their possible values better suit the use of equality (i.e., *Enum* or *String* values), which simplifies the rule's logic and execution time of the algorithm.

In our detection rules, only the values associated with a category in the genetic representation are subject to mutation. This approach helps maintain the rules' clarity and effectiveness.

```
1 if(duration <= 5855 AND protocolType == icmp AND
2 service == ecr_i AND flag == SF AND srcBytes >= 462 AND
3 dstBytes <= 8249 AND land == 0 AND wrongFragment <= 2 AND
4 urgent <= 0 AND count >= 1 AND srvCount >= 1 AND
5 serrorRate <= 0.81 AND srvSerrorRate <= 0.27 AND
6 rerrorRate <= 0.88 AND srvRerrorRate <= 0.31 AND
7 sameSrvRate <= 1.00 AND diffSrvRate <= 0.90 AND
8 srvDiffHostRate <= 1.00) {attack found}
```

Listing 1: DoS Detection Rule

Listing 1 shows an example of a DoS detection rule. A DoS attack is identified when all the criteria in the if condition are satisfied (e.g., "duration" ≤ 5855 suggests that one symptom of a potential DoS attack is characterized by a relatively short connection time

Table 2: Comparison with Al-fuhaidi et al. [1] for DoS attacks

Metrics	Presented GA	Al-fuhaidi et al.GA
Precision	0.993	0.999
Recall	0.999	0.998
F-Measure	0.996	0.999
Accuracy	0.994	0.999
Specificity	0.974	0.999
MCC	0.982	0.999
False Alarms	0.025	3e−5

between the host and server nodes). Similar considerations can be made for the other detection rules identified.

Table 1 shows the performance of the GA as the number of considered detection rules increases. The *"#Rules"* column indicates the number of rules considered. Out of the 17 computed rules, the first 11 are related to DoS attacks, the 12th to 15th are related to Probe attacks, and the last two are related to U2R attacks.

Recall, F-Measure, Accuracy, and Matthews Correlation Coefficient (MCC) performance improves as the number of rules considered increases. Similarly, the *False Alarm Rate (FA)* worsens. Finally, we added new detection rules related to Probe and U2R attacks compared to Al-fuhaidi et al. [1]. These rules increase the diversity of detectable attacks from one to four while increasing the performance of GA. While evaluating the GA's performance, as the number of detection rules considered increased, we realized that the first 15 rules were sufficient to detect the attacks for which they were developed and the R2L attack. This suggests a correlation between DoS and Probe attacks and R2L attacks that should be investigated in future work.

5.2 RQ_1. Using GAs to Detect DoS Attacks

To answer $\mathbf{RQ_1}$, we compared our results with those obtained by Al-fuhaidi et al.[1], as shown in Table 2. Both models have a high level of performance for all considered metrics. Our solution exhibits an increased recall compared to the original work, which allows it to detect a greater variety of attacks. However, other evaluation metrics show slight degradation. Observing the Specificity and False Alarm rate shows a clear difference between the performance of the detection rules generated in this study and the preliminary study. A lower score in these metrics indicates that the detection rules have a higher false positive rate.

Key findings for DoS attack detection. The detection rules for DoS attacks produced by GAs demonstrate performance similar to those derived by Al-Fuhaidi et al. [1]. Specifically, they exhibit higher recall rates despite being generated from a model that accounts for various attack types.

Table 3: Comparison with classifiers for DoS attacks

Metrics	GA	J48	BayesNet	SMO	NB	RF	DT
Precision	0.994	0.999	0.999	0.999	0.991	0.999	0.999
Recall	0.999	0.999	0.994	0.999	0.991	0.999	0.999
F-Measure	0.997	0.999	0.997	0.999	0.991	0.999	0.999
Accuracy	0.995	0.999	0.995	0.999	0.991	0.999	0.999
Specificity	0.974	0.999	0.999	0.999	0.999	0.999	0.998
MCC	0.984	0.999	0.985	0.998	0.972	0.999	0.999
False Alarms	0.026	1e−5	1e−5	1e−5	0.015	1e−5	0.001

Table 4: Comparison with classifiers for Probe attacks

Metrics	GA	J48	BayesNet	SMO	NB	RF	DT
Precision	0.620	0.997	0.814	0.993	0.318	0.999	0.979
Recall	0.987	0.995	0.987	0.980	0.936	0.996	0.986
F-Measure	0.761	0.996	0.892	0.986	0.475	0.997	0.983
Accuracy	0.975	0.999	0.990	0.998	0.916	1.000	0.999
Specificity	0.974	0.999	0.990	0.999	0.915	1.000	0.999
MCC	0.772	0.996	0.892	0.986	0.518	0.997	0.982
False Alarms	0.026	1e−5	1e−5	1e−5	1e−5	1e−5	1e−5

Table 5: Comparison with classifiers for U2R attacks

Metrics	GA	J48	BayesNet	SMO	NB	RF	DT
Precision	0.004	0.794	0.218	0.896	0.007	0.952	0.475
Recall	0.222	0.596	0.826	0.500	0.942	0.769	0.538
F-Measure	0.008	0.681	0.345	0.642	0.014	0.851	0.505
Accuracy	0.974	0.999	0.998	0.999	0.928	1.000	0.999
Specificity	0.974	0.999	0.998	1.000	0.928	1.000	1.000
MCC	0.028	0.688	0.424	0.669	0.077	0.856	0.505
False Alarms	0.026	1e−5	1e−5	1e−5	1e−5	1e−5	1e−5

5.3 $RQ_{2.}$ GAs Performance

Table 3 compares the performance of the GA and the classifiers for detecting DoS attacks. The performance of the GA can almost reach those of the ML models, particularly in terms of recall. Conversely, the specificity is lower, and the false alarm rate is slightly higher, denoting a higher tendency to generate false positives.

Table 4 shows the results obtained detecting *Probe* attacks. The GA method shows good potential despite a moderate precision of 0.620. The outcome is confirmed by observing the false alarm rate, which is equal to 0.026, higher than ML models. However, the GA showcases a recall value of 0.987, signifying a meager false-negative rate. Nevertheless, capturing the majority of positive instances remains a priority in security, even if it results in some false positives.

Table 5 shows the performance obtained in detecting *U2R* attacks. The results are strongly influenced by the unbalanced dataset, where high accuracy and specificity are obtained at the expense of the other metrics. As the specificity is higher than precision and recall, the set of detection rules generated can detect true negative instances. Furthermore, given the low number of positive class instances, the model cannot detect this type of instance, reporting a very low precision.

Table 6: Comparison with classifiers for R2L attacks

Metrics	GA	J48	BayesNet	SMO	NB	RF	DT
Precision	0.286	0.990	0.623	0.916	0.433	0.992	0.957
Recall	0.887	0.965	0.981	0.907	0.691	0.984	0.960
F-Measure	0.433	0.977	0.762	0.912	0.532	0.988	0.959
Accuracy	0.973	0.999	0.993	0.998	0.986	1.000	0.999
Specificity	0.974	0.999	0.993	0.999	0.990	1.000	1.000
MCC	0.495	0.977	0.779	0.911	0.540	0.988	0.958
False Alarms	0.026	1e−5	1e−5	1e−5	1e−5	1e−5	1e−5

Table 6 illustrates the results of detecting R2L attacks. Despite the dataset's imbalance due to the low occurrence of U2R attacks, the combination of detection rules targeting DoS and Probe attacks enhances the detection of R2L attacks, resulting in better performance in detecting R2L attacks. Specifically, the precision remains lower than ML models, with a value of 0.286. The Random Forest model exhibits the highest precision, scoring 0.992. However, while

we also discovered a low recall value for U2R attacks, the situation is different for detecting R2L attacks. The set of detection rules used and combined to detect R2L attacks presents a recall of 0.887, comparable to the one achieved by the ML model, with a maximum recall achieved by Random Forest with a score of 0.984. Therefore, detecting R2L attacks through the detection rules states a high false positive rate but with a high recall covering many attacks.

To conclude, while GAs obtain high performance to detect intrusion attacks, ML models outperform in this task for all of the four attacks analyzed. Unlike some ML models, which can obscure the rationale behind their detections, the clear conditions and thresholds set by GAs provide insights into how detections are made. This aids in investigating critical attack features and behaviours using conditions and thresholds defined in the detection rules. They serve as key elements for security testing, where these rules help develop test cases. These tests assess the robustness by emulating attack scenarios, confirming their efficacy in different contexts. While ML models perform better in detecting intrusion attacks, the detection rules provide a transparency and interpretability level that a tester can use to understand the conditions in which an attack occurs.

Key findings for GAs Performance. ML algorithms outperform GAs for detecting intrusion attacks, but detection rules allow the security tester to understand the conditions and thresholds in which the attack occurs.

6 THREATS TO VALIDITY

This section presents the main limitations of our studies.

Construct Validity. We are conscious that the dataset selection may influence the results. However, this decision enabled us to compare our results with the state-of-the-art and to contextualize them better. After several attempts, the best combination of encodings to use in cascade was 11 DoS, four Probe, and two U2R encoded rules.

External Validity. Given the imbalanced dataset, our results' generalizability may be reduced, affecting detection rules for U2R and R2L attacks. Therefore, we combined detection rules for Probe and DoS attacks to detect R2L attacks, enhancing detection capabilities. Future efforts will focus on collecting more data on Probe, U2R, and R2L attacks and expanding KDDCUP99 by integrating multiple datasets from the same domain. Another critical concern is the risk of overfitting GAs, leading to detection rules that may only fit part of the dataset. To mitigate this, we introduced multiple detection rules for each attack and diversified conditions for intrusion detection, enhancing robustness.

Conclusion Validity. To evaluate the performance of our detection rules and compare them to the work of Al-Fuhaidi et al.[1], we adopted the performance metrics they used. However, relying solely on these metrics can provide a limited understanding of the detection rules' capabilities. To address this, we expanded the set of metrics including F-measure, Recall, Specificity, and MCC, enabling us to assess the performance of detection rules better.

7 CONCLUSIONS AND FUTURE WORK

This study evaluated the adoption of GAs to create 17 detection rules to identify four attack types. The results denote good performance in detecting intrusion attacks when the adopted dataset features an adequate number of malicious instances. Findings of the study open the possibility of having explainable solutions to generate test cases based on the features adopted to create the detection rules.

Future work entails applying and validating these features for test case generation in industrial contexts. Furthermore, we will investigate the characteristics of R2L attacks, finding correlations with other types of attacks based on the features that we selected.

CREDITS

The work is an extended version of Davide La Gamba's Bachelor thesis, developed at the University of Salerno in 2022.

REFERENCES

[1] Belal Al-fuhaidi, I Abd-Alghafar, Gouda Salama, and A Abd-Alhafez. 2009. Performance Evaluation of a Genetic Algorithm Based Approach to Network Intrusion Detection System.
[2] Safaa O Al-mamory and Firas S Jassim. 2013. Evaluation of different data mining algorithms with kdd cup 99 data set. *Journal of Babylon University/Pure and Applied Sciences* 21, 8 (2013), 2663–2681.
[3] Mohammed Aziz Al Kabir, Wael Elmedany, and Mhd Saeed Sharif. 2023. Securing IoT devices against emerging security threats: challenges and mitigation techniques. *Journal of Cyber Security Technology* 7, 4 (2023), 199–223.
[4] Oluleye H Babatunde, Leisa Armstrong, Jinsong Leng, and Dean Diepeveen. 2014. A genetic algorithm-based feature selection. (2014).
[5] Datta H Deshmukh, Tushar Ghorpade, and Puja Padiya. 2014. Intrusion detection system by improved preprocessing methods and Naïve Bayes classifier using NSL-KDD 99 Dataset. In *2014 International Conference on Electronics and Communication Systems (ICECS)*. IEEE, 1–7.
[6] Dario Di Dario, Valeria Pontillo, Stefano Lambiase, Filomena Ferrucci, and Fabio Palomba. [n. d.]. Security Testing in The Wild: An Interview Study. In *2023 49th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE.
[7] Giammaria Giordano, Fabio Palomba, and Filomena Ferrucci. 2022. On the use of artificial intelligence to deal with privacy in IoT systems: A systematic literature review. *Journal of Systems and Software* 193 (2022), 111475.
[8] Gunes Kayacık, A. Zincir-Heywood, and Malcolm Heywood. 2005. Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99.
[9] Dong Seong Kim, Ha-Nam Nguyen, and Jong Sou Park. [n. d.]. Genetic algorithm to improve SVM based network intrusion detection system. 155–158 vol.2.
[10] Davide La Gamba, Gerardo Iuliano, Gilberto Recupito, Giammaria Giordano, Filomena Ferrucci, Dario Di Nucci, and Fabio Palomba. 2024. Web Appendix of the paper. https://figshare.com/s/f7fc3d84dac00410bab3. Online.
[11] Yuan-Cheng Li and Zhong-Qiang Wang. 2007. An intrusion detection method based on SVM and KPCA. In *2007 International Conference on Wavelet Analysis and Pattern Recognition*, Vol. 4. 1462–1466.
[12] Tao Lu, Youpeng Huang, Wen Zhao, and Jie Zhang. 2019. The metering automation system based intrusion detection using random forest classifier with smote+enn. In *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*. IEEE, 370–374.
[13] Ibrahim Obeidat, Nabhan Hamadneh, Mouhammd Alkasassbeh, Mohammad Almseidin, and Mazen AlZubi. 2019. Intensive pre-processing of kdd cup 99 for network intrusion classification using machine learning techniques. (2019).
[14] Patrick M. Reed, Barbara S. Minsker, and David E. Goldberg. [n. d.]. *The Practitioner's Role in Competent Search and Optimization Using Genetic Algorithms.*
[15] Suchet Sapre, Pouyan Ahmadi, and Khondkar Islam. 2019. A robust comparison of the KDDCup99 and NSL-KDD IoT network intrusion detection datasets through various machine learning algorithms. *arXiv preprint arXiv:1912.13204* (2019).
[16] Mohammad Khubeb Siddiqui and Shams Naahid. 2013. Analysis of KDD CUP 99 dataset using clustering based data mining. *International Journal of Database Theory and Application* 6, 5 (2013), 23–34.
[17] Gary Stein, Bing Chen, Annie S. Wu, and Kien A. Hua. 2005. Decision tree classifier for network intrusion detection with GA-based feature selection *(ACM-SE 43)*. New York, NY, USA, 6 pages.
[18] Ravindra Gupta Swati Paliwal. 2012. Denial-of-Service, Probing & Remote to User (R2L) Attack Detection using Genetic Algorithm. *International Journal of Computer Applications* 60, 19 (December 2012), 57–62. https://doi.org/10.5120/9813-4306
[19] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 1–6.
[20] John R Vacca. 2012. *Computer and information security handbook*. Newnes.
[21] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science.
[22] Bruno Bogaz Zarpelão, Rodrigo Sanches Miani, Cláudio Toshio Kawakani, and Sean Carlisto de Alvarenga. 2017. A survey of intrusion detection in Internet of Things. *Journal of Network and Computer Applications* 84 (2017), 25–37.