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

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

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

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

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

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

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

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

ABSTRACT

Internet-of-Things (IoT) devices generate and exchange vast amounts of data daily, presenting significant challenges to security, privacy, and safety due to insecure data exchange mechanisms. Security testing, leveraging machine learning (ML), offers a promising avenue to identify and classify potential malicious network traffic. Prior studies have suggested that ML solutions can inform security testers on designing testing activities for IoT malicious network traffic attacks. In this paper, we propose a search-based approach using Genetic Algorithms (GAs) to evolve detection rules and identify intrusion attacks, offering insights into designing security tests. Our contribution includes defining 17 rules for intrusion attack detection, extending previous work on GAs for intrusion detection, and comparing GAs with state-of-the-art ML models. Our findings demonstrate that GAs can effectively replace ML algorithms with sufficient attacking examples, maintaining performance while enhancing usability and facilitating the 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.

SECUTE, June 18–21, 2024, Salerno, Italy

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Davide La Gamba, Gerardo Iuliano, Gilberto Recupito, Giammaria Giordano, Filomena Ferrucci, Fabio Palomba, and Dario Di Nucci. 2018. Toward a Search-Based Approach to Support the Design of Security Tests for Malicious Network Traffic. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (SECUTE).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Internet-of-Things (IoT) devices interact with people or govern environments. According to Statista.com,[1] the number of connected IoT devices will be close to 19 billion by 2025 and is expected to reach 29 billion by 2030. This rapid growth has been boosted by their relatively low cost, small dimensions, and capability to be adapted and used in multiple contexts for general-purpose tasks [6]. IoT devices produce and exchange data daily, inside their work environment and through user interactions [3].

Regrettably, not all that glitters is gold: the sheer volume of data generated and exchanged by IoT devices raises significant concerns regarding security and testing, particularly in the realm of network security. This data, vulnerable to intrusion by external malicious attackers, underscores the critical need for robust security measures and thorough testing protocols to safeguard against potential threats. On the one hand, the data produced by IoT devices need to be stored and exchanged by adopting secure protocols to limit possible malicious attacks that could compromise the users' security and privacy [18]. On the other hand, testing activities can support practitioners in discovering bugs or system weaknesses that malicious users could exploit to conduct an attack [16].

Due to the extensive adoption of IoT devices worldwide, the software engineering (SE) community primarily focused on analyzing how *security testing* can help to prevent potential attacks [8] and support practitioners to improve the robustness of the system to defend from malicious attacks.

Recent research by Giordano et al. [10] conducted a systematic literature analysis of the integration of artificial intelligence (AI)

[1] https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

techniques to address privacy and security challenges in IoT systems. Their analysis revealed a prevalent trend wherein AI methodologies, particularly machine learning (ML), are utilized within the domain of security testing. One notable application of AI algorithms in security testing is their potential to provide a foundational framework for the design of security test cases. Specifically, these algorithms can offer valuable insights that enable security testers to craft effective test cases. In the specific context of IoT network security, intrusion detection models serve as a crucial starting point for supporting the design of security tests. These models employ specific conditions and thresholds to identify and classify intrusion attacks. By leveraging the outputs of intrusion detection models, security testers can establish the criteria and parameters necessary to define comprehensive security test cases tailored to address potential vulnerabilities and threats in IoT networks.

Al-fuhaidi et al. [1] employed Genetic Algorithms (GAs) to define detection rules for identifying potentially malicious traffic, focusing on the KDDCUP99 dataset [30], which includes four network attack categories: DoS, Probe, U2L, and R2L. 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 Genetic Algorithms (GAs), to define detection rules for identifying potentially malicious traffic, their focus was primarily on DoS attacks using the KDDCUP99 dataset [30]. In our paper, we build upon this work by providing two significant extensions. Firstly, we expand the scope beyond DoS attacks to cover all attack categories present in the KDDCUP99 dataset. This extension involves generating specific detection rules tailored to define the properties and characteristics of each malicious attack category, thereby enabling a more comprehensive identification of various types of network intrusions. Secondly, we enhance the comparison with machine learning (ML) algorithms by introducing three additional ML techniques not previously considered. By including a broader range of ML methods, we aim to provide a more thorough evaluation of the effectiveness of our set of detection rules compared to ML techniques.

These contributions are expected to advance the field of security testing in three significant ways: (i) By establishing a one-to-one mapping between each detection rule and a specific attack category, our approach empowers security testers to develop secure test cases capable of effectively identifying individual attack types. (ii) The adoption of multiple detection rules facilitates maintenance and evolutionary activities in security testing, enabling practitioners to modify detection rules for a specific attack without impacting on the detection performance of the detection rules for other attacks. (iii) By associating each detection rule with a single attack, our approach has the potential to reduce false positives and negatives, thereby enhancing the accuracy and reliability of security testing.

To sum up, our work provides 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 [30].
- A comparison between genetic algorithms and machine learning models to identify malicious traffic, which practitioners and security testers can use.
- A publicly available replication package [23] with both data and scripts used in the study, which researchers can use to build upon our findings and/or replicating our results.

Structure of the Paper. Section 2 provides details on the background information. Section 3 summarizes and contrasts the state-of-the-art literature. Section 4 describes the research method applied to conduct our experiments and defines the research questions of the study. Section 5 shows the results achieved and provides insight for researchers and practitioners. Section 6 describes the main limitations of our study and how we mitigated them. Lastly, Section 7 concludes the paper and provides some final remarks.

2 BACKGROUND

This section analyzes the main characteristics of IoT devices and discusses the uses of artificial intelligence in solving these issues.

2.1 Intrusion Detection

Intrusion detection involves identifying unauthorized actions against information systems, known as intrusions, aimed at gaining unauthorized access to a computer system [33]. 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 [31]. Such activities pose various threats to network security, ranging from *Denial of Service* (DoS) attacks to *unauthorized access attempts* [21]. DoS attacks, for instance, disrupt normal system functioning by overwhelming resources, making the system inaccessible to legitimate users through techniques like Apache, Smurf, and Ping of Death. *Intrusion Detection Systems* (IDS) are instrumental in safeguarding network integrity against such threats, monitoring system interactions for signs of malicious behavior indicative of an ongoing attack. IDS utilizes different data sources, such as Network-Based IDS (NIDS) analyzing network traffic and Host-Based IDS (HIDS) monitoring individual host devices for compromise indicators. Organizations may prioritize NIDS for its centralized monitoring, comprehensive coverage, scalability, and real-time threat detection capabilities, especially in environments with critical concerns about external threats and network-wide visibility [33]. Exploiting data sources collected and available for research based on network traffic creates the opportunity to use machine learning to detect such intrusions [4]. Therefore, popular datasets containing information on the traffic with specific attacks are used to train ML models. *KDDCUP99* [30] is one of the most popular datasets in this context. ML models using this dataset show high performance, implying that the information in the dataset, i.e., the protocol type and the parameters of each communication in the network traffic, is essential to detect intrusions. More details concerning the structure of this dataset are reported in Section 4.1. This study extends a previous work [2] to exploit the *KDDCUP99* dataset using a GA-based method.

2.2 Genetic Algorithm Applications

Genetic Algorithms (GAs) were introduced by Holland and their students in the 1960s [20]. The base idea of these algorithms is to conceptually "translate" the *natural selection* process described in the Darwin theory in computer science algorithms to evolve candidate solutions, namely "*population of individuals*" applying

genetic-inspired procedures of *Crossover* and *Mutation* to identity optimal solutions in a search space [19].

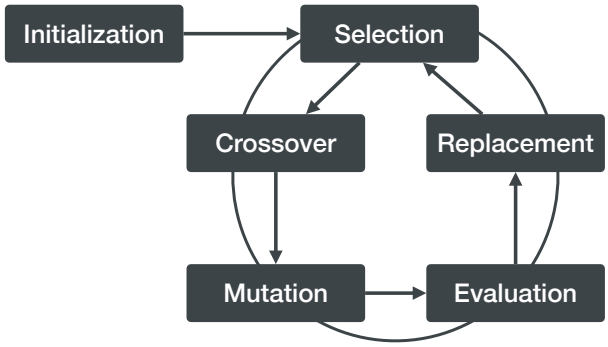

Figure 1: A Conceptual Implementation of a Genetic Algorithm model.

Figure 1 illustrates the conceptual implementation of a GA. As depicted in the figure, the process initiates with the *initialization* step, where an initial population of candidate solutions is generated. Subsequently, *Selection* occurs, involving the selection of a group of parents from the previous generation based on their assigned fitness values. A *fitness function* assesses the quality of potential solutions by evaluating their suitability for the given problem. *Crossover* then comes into play, merging the genetic data of two parents to generate new offspring. Similarly, *Mutation* maintains genetic diversity, mirroring biological mutation processes. Following these steps, *Evaluation* takes place, where the modified population is evaluated using the fitness function. Finally, *Replacement* happens as the current population replaces the old one, marking the beginning of a new process iteration [20].

The application of GAs is largely explored in software testing of traditional systems due to their versatility and adaptability [9, 11, 17, 26], whereas its application to detect intrusion attacks is still on a preliminary state [2].

3 RELATED WORK

We summarize how AI-based methods were used to detect intrusion attacks leveraging the KDDCUP99 dataset.

Stein et al. [28] applied GAs to perform a feature selection task. The most relevant features identified by the GA were provided to a Decision Tree classifier. Their results revealed that such feature selection can boost the performances of the models.

Dong et al. [13] and Li and Wang [14] improved SVM-based models by incorporating GAs. Their results corroborate previous studies, showing that pre-processing operations and feature selection drastically increase the model performances.

Compared to the clusters of studies discussed above, our approach aims to use GAs to identify detection rules to discover potential malicious network traffic without relying on ML.

Looking closely at studies identifying detection rules, Paliwal and Gupta [29] applied GAs to identify an omni-comprehensive detection rule to identify all the possible attacks in the KDDCUP99 dataset. The results show a detection rate of 97% on all attack instances. The differences between Paliwal and Gupta's work are twofold: (i) we consider multiple detection rules; (ii) we compare our results to several state-of-the-art ML algorithms.

To our knowledge, the closest work was conducted by Al-fuhaidi et al. [1], who used GA to identify detection rules for DoS attacks and compared their approach with three ML techniques i.e., Bayes Network, Decision Tree, and SVM. Their results achieved a detection rate of up to 99% and a false positive rate of 0.003%.

In the context of this paper, we replicate the method applied by Al-fuhaidi et al. to generate a set of detection rules able to identify all the possible categories of attacks available on the KDDCUP99 dataset and extend the number of ML algorithms by including not only those considered by Al-fuhaidi et al. but also other three algorithms i.e., Decision Table, Naive Bayes, and Random Forest.

4 RESEARCH METHOD

The *goal* of this study is to analyze to what extent GA algorithms can be used to detect malicious network traffic provided by IoT devices. To achieve this goal, we generated a set of rules "*if (condition) then {action}*" i.e., ***if*** all the conditions that compose a specific rule are respected, ***then*** a specific attack is detected. The *purpose* is to provide a set of rules that security testers can use to build security tests. The *quality focus* is on GA algorithms 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 the use of GA to discover suspect network traffic. At the same time, the latter are interested in discovering automatic solutions to implement more deterministic tests. The *context* of our investigation is the *KDDCUP99* [30] dataset, featuring four categories of attacks: DoS, Probe, U2R, and 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$_2$*** *To what extent can genetic algorithms generate detection rules to identify other attack types?*

After assessing the performance of the models in detecting DoS attacks, we consider the other attack categories available in the KDDCUP99 dataset. To improve the understandability and the readability, we split **RQ**$_2$ into three different sub-research questions according to the specific network attack identified by the authors of KDDCUP99; we asked:

> 🔍 ***RQ$_{2.1}$*** *To what extent can genetic algorithms generate detection rules to identify Probe attacks?*

> 🔍 ***RQ$_{2.2}$*** *To what extent can genetic algorithms generate detection rules to identify U2R attacks?*

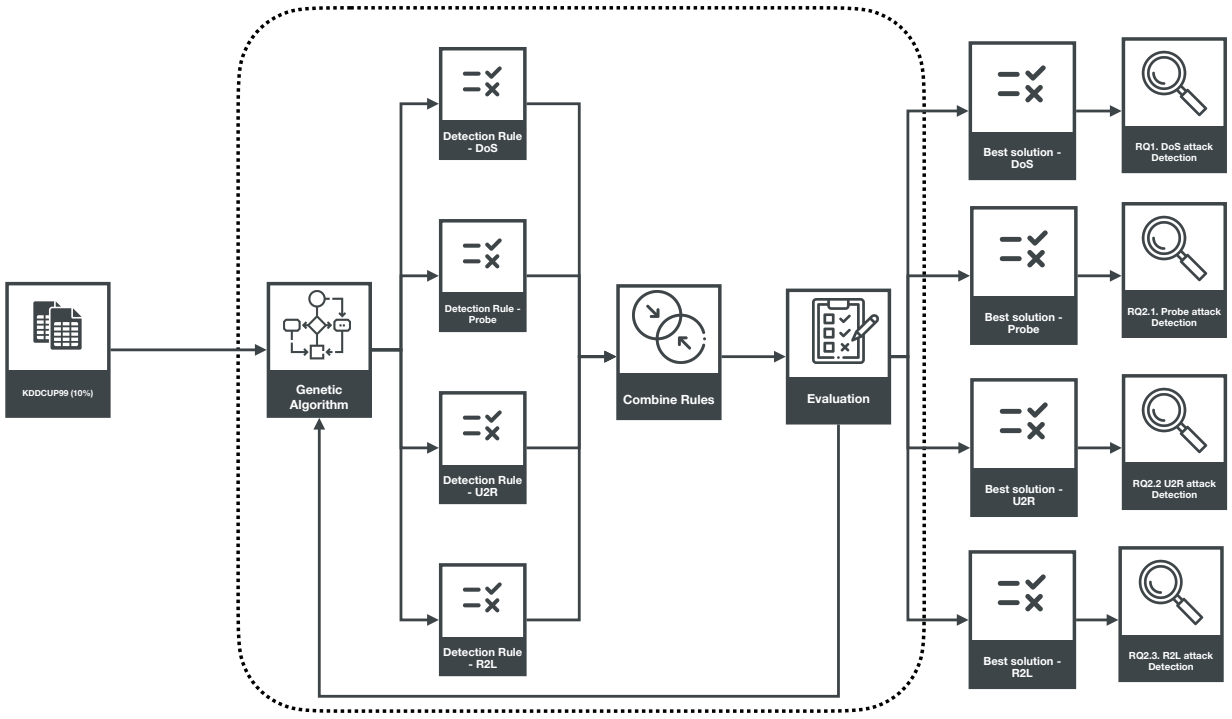

Figure 2: The Research Method Applied for our Study.

> 🔍 ***RQ$_{2.3}$*** *To what extent can genetic algorithms generate detection rules to identify R2L attacks?*

We design the study following the guidelines by Wohlin et al. [32] and the *ACM/SIGSOFT Empirical Standards*.[2] Figure 2 illustrates the research process we applied to perform our investigation. First, we select a large and well-established dataset that contains information on network attacks in IoT environments, i.e., KDDCUP99. Then, we develop a GA algorithm configuring the population size at 1,000 individuals and with 1,000 generations to extract the detection rules to identify network attacks. In each generation, (i) we extract the detection rule for each category, (ii) we combine the detection rules, (iii) we evaluate the combined rules and compare them with the previous generation and retain the best ones. At the end of the process, we use the final detection rules to evaluate the GA performances for each attack.

4.1 Dataset Description

To achieve our objective, we select the KDDCUP99 dataset i.e., a dataset concerning the connections in a military-grade network. The motivations to select this dataset are fourfold: (i) the dataset is one of the largest dataset publicity 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 largely analyzed in previous work resulting on the top-3 of the most used dataset according to the SLR conducted by Giordano et al. [10], (iii) Al-fuhaidi et al. [1] leverages this dataset to perform its analysis, and (iv) KDDCUP99 features five categories of network traffic [30]:

Normal refers to network connections considered legitimate.

Denial of Service Attack (DoS) refers to an assault wherein the attacker overwhelms a computing or memory resource, rendering it too occupied or overloaded to manage genuine requests, consequently denying legitimate users access to the machine.

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 where the attacker initially possesses access to a standard user account on the system, potentially acquired through methods such as password sniffing, dictionary attacks, or social engineering. Subsequently, they exploit a vulnerability to escalate their privileges to root access on the system.

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.

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, 25, 27].

To analyze the dataset characteristics, we preliminary analyzed the number of instances for each network traffic label (i.e., DoS, Normal, Probe, R2L, and U2R) to analyze their distributions. Figure 3

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

overviews the number of connection records on the 10% of the dataset labeled according to their category.

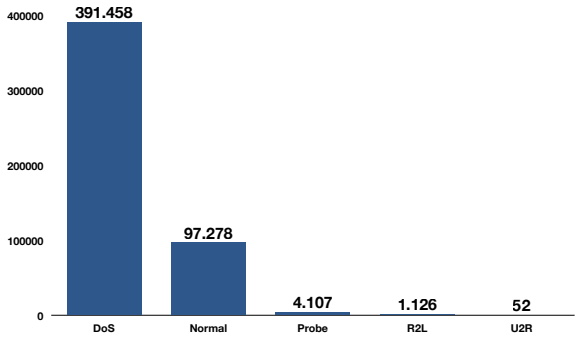

Figure 3: Number of connection records labeled by category.

As can be observed from the figure, out of a total of over 494k connection instances, 79% of them are classified as DoS attacks, 20% as Normal traffic, while the remaining 1% are divided among the other three categories (Probe, R2L, and U2L) i.e., the dataset is highly unbalanced.

4.2 Encoding of the Individuals

The individual in the population is a set of *if (condition) then {action}* rules, aiming at detecting harmful connections. Specifically, the *condition* represents the trigger of the generated rule, while the *action* is a warning concerning the harmful connection or an action for analysis purposes. Such parts of the chromosome are connected using the logical AND operator, while the verses of the inequalities to be checked for each feature are encoded, thus subject to evolution, except for genes referring to categorical features, for which equality is always used. Each numerical gene of the chromosome is an *Integer* value bound by a minimum and a maximum, specific to the related feature its rule applies to. These values were extrapolated following a study of the respective feature boundaries in the dataset so as not to explore potentially irrelevant regions of the solution space. Nevertheless, this approach may hinder the ability to classify connections with feature values failing outside the limits extrapolated from the dataset. We provide an encoding for each of the four attack categories, each with a different number of genes based on the most relevant features for the specific network attack. We considered the features obtained by Kayacık et al. [12] leveraging the *Information Gain* algorithm.

DoS Encoding. This chromosome consists of 11 parts, seven of which contain exclusively one gene, as these are linked to features with unique minimum and maximum values compared to the others. The only features with equal values are "WrongFragment", "Urgent", "Count" and "SrvCount", and those expressing percentage values. The last chromosome contains the genes related to the verses of the inequalities to be tested. Specifically, "1" represents "$\leq$" while "2" represents "$\geq$". Equality signs were left out because, in the initial phase of attempts, they did not prove to classify continuous features correctly and thus would have slowed down the process of evolving the rules. Table 1 shows the features identified with the minimum and maximum values for detecting DoS attacks.

Table 1: Extreme values feature DOS solution.

Feature	(Min, Max)
Duration	(0, 58329)
ProtocolType	(1, 3)
Service	(1, 66)
Flag	(1, 11)
SrcBytes	(0, 999)
DstBytes	(0, 9999)
Land	(0, 1)
WrongFragment	(0, 3)
Urgent	(0, 3)
Count	(0, 99)
SrvCount	(0, 99)
SerrorRate	(0, 100)
SrvSerrorRate	(0, 100)
RerrorRate	(0, 100)
SrvRerrorRate	(0, 100)
SameSrvRate	(0, 100)
DiffSrvRate	(0, 100)
SrvDiffHostRate	(0, 100)

Table 2: Extreme values feature Probe solution.

Feature	(Min, Max)
Duration	(0, 58329)
DstBytes	(0,9999)
LoggedIn	(0,1)
SuAttempted	(0,2)
NumShells	(0,2)
NumRoot	(0, 993)
NumFilesCreations	(0, 28)
NumAccessFiles	(0, 8)
SrvDiffHostRate	(0, 100)
DstHostSrvDiffHostRate	(0, 100)
RerrorRate	(0, 100)
DstHostCount	(0, 260)
Flag	(1, 11)

Probe Encoding. Through the analysis conducted by Kayacık et al. [12], the features shown Table 2 were considered, along with their respective ranges of values. For some features, upper bounds were chosen slightly higher than those in the dataset to detect network connections with slightly higher values.

U2R Encoding. We used the same encoding strategy described for *DoS* and *Probe* attacks. The selected features and their ranges of values are shown in Table 3.

R2L Encoding. The encoding comprises a combination of DoS and Probe encodings, already described previously.

Table 3: Extreme values feature U2R solution.

Feature	(Min, Max)
Duration	(0, 58329)
SrcBytes	(0, 999)
DstBytes	(0,9999)
LoggedIn	(0,1)
SuAttempted	(0,2)
NumShells	(0,2)
NumRoot	(0, 993)
NumFilesCreations	(0, 28)
NumAccessFiles	(0, 8)
SrvDiffHostRate	(0, 100)
DstHostCount	(0, 260)

4.3 Fitness Function

To compare our results with the original study, we adopt the same fitness function provided by Al-fuhaidi et al. [1]. We report the formula used with a quick explanation of their variables.

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

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.

4.4 Parameters

We attempt several parameter configurations when developing the algorithm, such as the number of individuals, termination criterion, crossover and mutation probability, and related operators. The final configuration consists of 10% and 90% mutation and crossover probability. The selection operator is the *EliteSelector* [5] to ensure a form of **elitism** [24] in the population, allowing the best solutions to survive to the next generation without change. The mutation operator is the default operator, which mutates the genes of individuals by obtaining new values for the predetermined ranges to better explore possible solutions by ensuring greater diversity. The population size of individuals is set at 1,000, with a stopping criterion set at 1,000 generations.

Table 4: Differences between the ML techniques used by Al-fuhaidi et al. and our work.

Machine Learning Technique	Al-fuhaidi et al.	Our Work
Bayes Network	●	●
Decision Table	●	●
Decision Tree (J48)	●	●
Naive Bayes	●	●
Random Forest	●	●
Support Vector Machine (SMO)	●	●

4.5 Validation

We compare the performance of our method with a set of well-known 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. 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, 7, 15, 22]. Table 4 summarizes the performance differences obtained by the original work and ours; specifically, the circles indicate the applied ML techniques, i.e., the green circles represent those applied, and the red circles depict those not applied.

5 ANALYSIS OF THE RESULTS

This section describes the results of our work. We describe the detection rules, how they were developed, and the decisions made about them. Next, we show the results for each RQ.

5.1 Detection Rules

We developed 17 detection rules: 11 for DoS attack detection, four for Probe attacks, and two for U2R attacks. A concatenation of DoS and Probe rules was used to discover R2L attacks. Thus, each detection rule comprises the features most relevant for the attack to be detected, according to Kayacık et al. [12]. For the *ProtocolType*, *Service*, *Flag*, and *Land* features, we did not use the greater equal and less equal operators but only the equal operator. The reason lies in the type of features, the values they can take, and the running time of the algorithm. The detection rules for DoS and Probe attacks are also relevant to detecting R2L attacks. In the following, we show how some generated rules are structured.

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

Listing 1: DoS Detection Rule

```
if(duration <= 3290 AND dstBytes <= 5389 AND
loggedIn >= 0 AND suAttempted <= 2 AND numShells <= 0 AND
numRoot <= 184 AND numFilesCreations <= 2 AND
numAccessFiles <= 6 AND srvDiffHostRate <= 1.00 AND
dstHostSrvDiffHostRate <= 0.53 AND rerrorRate <= 1.00 AND
dstHostCount >= 13 AND flag == S0) {attack found}
```

Listing 2: Probe Detection Rule

```
if(duration <= 26427 AND srcBytes <= 6 AND
dstBytes <= 2731 AND loggedIn <= 1 AND
suAttempted <= 0 AND numShells <= 1 AND
numRoot <= 648 AND numFilesCreations <= 4 AND
numAccessFiles <= 2 AND srvDiffHostRate <= 57 AND
dstHostSrvDiffHostRate <= 2) {attack found}
```

Listing 3: U2R Detection Rule

Listing 1 shows an example of a computed 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 between the host and server nodes). Similar considerations can be made for the other detection rules identified.

Table 5 shows the performance of the GA as the number of considered detection rules increases. 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 MCC performance improve as the number of rules considered increases. Similarly, the *False Alarm Rate* worsens; however, from a Security Testing perspective, it remains an acceptable value, suggesting that the system effectively detects genuine attacks without overly increasing the number of false positives. Finally, we added new detection rules compared to 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 the genetic algorithm.

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]. The results are shown in Table 6.

Both models have a high level of performance for all considered metrics. In addition to detecting a greater variety of attacks, the proposed solution exhibits an increased recall compared to the original work, although other evaluation metrics show slight degradation. A clear difference between the performance of the detection rules generated in this study and the preliminary study can be seen by observing the specificity and the false Alarms Rate. A lower score in these metrics indicates that the detection rules have a higher false positive rate. In conclusion, the results align with those achieved by Al-Fuhaidi et al. [1].

Table 7 compares the performance of the GA and the classifiers for detecting DoS attacks. The performance of the GA can almost reach those of the machine learning models, particularly in terms of recall. Conversely, the specificity is lower, and the false alarm rate is slightly higher, denoting a higher tendency of the detection rules to generate false positives.

Key findings for RQ_1

The detection rules for Denial of Service (DoS) attacks produced by the GA demonstrate performance similar to those derived by Al-fuhaidi et al. Specifically, they exhibit higher recall rates despite being generated from a model that accounts for various attack types

5.3 $RQ_{2.1}$. Using GAs to Detect Probe Attacks.

Table 8 shows the results obtained detecting *Probe* attacks. The GA method shows good potential in detecting probe attacks despite a moderate precision of 0.620, indicating a high false positive rate for this type of attack. The outcome is confirmed by observing the false alarm rate, which is equal to 0.026, higher than the false alarm rate of the 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.

Key findings of $RQ2_1$

The detection rules to identify Probe Attacks exhibit a notable high recall and a minimal false-negative rate. However, the precision is not equally high, suggesting the presence of false positives in the detection process.

Table 9: Comparison of presented Genetic Algorithm with classifiers for U2R attack.

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

5.4 $RQ_{2.2}$. Using GAs to Detect U2R Attacks.

Table 9 shows the performance obtained in detecting *U2R* attacks. The results are strongly influenced by the unbalanced dataset. Indeed, high accuracy and specificity are obtained at the expense of the other metrics. As the specificity is higher than precision and recall, we note that the set of detection rules generated can detect true negative instances (i.e., the network connections classified as not malicious). Furthermore, given the low number of instances of the positive class (i.e., the network connections that are classified as "*U2R*"), the model cannot detect this type of instance, reporting a very low precision.

Key findings of $RQ2_2$

Despite exhibiting high accuracy and specificity, the performance of the detection rules obtained by the GA is negatively affected by the data imbalance, resulting in weaker precision in detecting U2R attacks.

5.5 $RQ_{2.3}$. Using GAs to Detect R2L Attacks.

Table 10 illustrates the results obtained in detecting R2L attacks. The performance detecting R2L attacks yields better results than the detection proposed for U2R attacks. Despite the imbalance of the dataset caused by the low presence of these types of attacks, which clearly affects the detection performance, the combination of detection rules aimed at detecting DoS and Probe attacks improves the detection of 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,

Table 5: Performance of the GA as rules increase.

		Performance Metrics						
Rule Type	# Rules	Precision	Recall	F-Measure	Accuracy	Specificity	MCC	False Alarm 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

Table 6: Comparison of presented Genetic Algorithm with Al-fuhaidi et al. Genetic Algorithm for detecting 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

Table 7: Comparison of presented Genetic Algorithm with classifiers for DoS attack

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

comparable to the one achieved by the ML model, with a maximum recall achieved by Random Forest with a score of 0.984.

In conclusion, detecting R2L attacks through the detection rules generated by GA clearly states a high false positive rate but with a high recall that covers a high number of R2L attacks.

Table 8: Comparison of presented Genetic Algorithm with classifiers for Probe attack.

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 10: Comparison of presented Genetic Algorithm with classifiers for R2L attack.

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

Key findings of $RQ2_3$

The precision of the detection rules for R2L attacks remains lower than ML models, combining detection rules for DoS and Probe attacks. However, it significantly enhances the recall for R2L attacks, effectively covering many instances despite a high false positive rate.

5.6 Genetic Algorithm vs ML Models

GAs exhibit lower accuracy than ML models; therefore, adopting GAs in IDSs requires careful consideration. Specifically, when datasets contain a sufficient number of positive instances (i.e., for DoS and Probe attacks), GAs demonstrate a comparable and effective capability in identifying intrusion attempts.

However, the efficacy of GAs in generating detection rules for attack types characterized by limited instances, e.g., R2L and U2R attacks, is often hindered by challenges arising from the limited availability of data. Hence, expanding datasets to include more examples of these less frequent attack types is essential to refine GAs detection accuracy.

The transparency and explainability of the generated detection rules are crucial motivators for adopting GAs. Unlike some ML models, in which the motivation behind the detection can be challenging to extract, the conditions and thresholds established by GAs provide clear insights into the detection process. This transparency enhances understanding and facilitates the identification of key attack characteristics and behaviors. The conditions and thresholds established for each detection rule can be used as base elements to understand the properties of intrusion attacks. This step is essential to conducting rigorous testing activities, wherein the detection rules provide a starting point for generating diverse test cases. These tests evaluate the robustness of an IDS by simulating various intrusion scenarios, ensuring its effectiveness across different attack contexts.

6 THREATS TO VALIDITY

This section presents the main limitations of our studies and discusses the applied mitigation strategies.

Construct Validity. This threat regards the differences between the *theory* and the *observation*. The main concern regards the dataset used for our investigation i.e., KDDCUP99. We are conscious that the dataset selection may have influenced the results. However, this decision enabled us to compare our results with the state-of-the-art and, in turn, to contextualize them better.

Another essential aspect to consider was the order of application of the different algorithms with the three codings reported in section 4.2. After several attempts, the best combination of encodings to use in cascade was 11 DoS, four Probe, and two U2R encoded rules.

External Validity. A threat to external validity is related to the number of observations for some attack types. Given the unbalanced nature of the dataset, the reduced generalizability of our results can affect some detection rules related to U2R and R2L. To mitigate this threat, we combined the detection rules related to Probe and DoS attacks to detect R2L attacks, ensuring this attack could be detected. In the future, we plan to address the issue of U2R attacks, collecting new information on this type of attack and extending the number of observations in KDDCUP99 (e.g., combining multiple datasets on the same domain).

Another critical threat is the risk of overfitting genetic algorithms, which creates a specific detection rule that overfits only part of the dataset. To address this concern, we considered, starting from the individual set, more detection rules for a single attack and the ability to detect intrusion attacks using different conditions. Combining more rules allows for a more robust detection.

Another critical issue is the selection of ML algorithms to conduct our comparison with the detection rules generated by the proposed genetic algorithm. While prior work by Al-fuhaidi et al. [1] provided a set of algorithms for comparison, our research extends this comparison by including three additional commonly used ML algorithms. This expansion enriches the benchmarking process, ensuring a more comprehensive performance evaluation of the detection rules. By incorporating a broader range of ML techniques, our study aims to offer a nuanced understanding of how the proposed genetic algorithm stacks up against established methods in detecting intrusion attacks.

Conclusion Validity. To evaluate the performance of our detection rules and to compare to the preliminary work of Al-Fuhaidi et al. [1], we considered the performance metrics that authors used in preliminary work (i.e., Accuracy, Detection Rate, and False Alarms). However, the exclusive use of these metrics can mislead the understanding of the capabilities of the detection rules. To address this threat, we extended the set of metrics, including F-measure, Recall, Specificity, and Matthews Correlation Coefficient (MCC). This expansion of metrics allows us to better highlight the tendency of detection rules to minimize false negatives in comparison to the work of Al-Fuhaidi et al. and machine learning models.

7 CONCLUSIONS AND FUTURE WORK

This study evaluated the adoption of GAs to create specific detection rules for intrusion attacks. We extracted 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. The set of the generated rules opens the possibility of having explainable solutions.

Future work entails replicating the study on additional datasets and industrial contexts to verify to what extent the results are generalizable. Furthermore, to understand the applicability of the extracted detection rules, we will evaluate the best properties and criteria that can be adopted for automatically generate security tests specific to IDSs.

REFERENCES

[1] Belal Al-fuhaidi, I Abd-Alghafar, Gouda Salama, and A Abd-Alhafez. Performance evaluation of a genetic algorithm based approach to network intrusion detection system. 05 2009.

[2] Safaa O Al-mamory and Firas S Jassim. Evaluation of different data mining algorithms with kdd cup 99 data set. *Journal of Babylon University/Pure and Applied Sciences*, 21(8):2663–2681, 2013.

[3] Eyhab Al-Masri, Karan Raj Kalyanam, John Batts, Jonathan Kim, Sharanjit Singh, Tammy Vo, and Charlotte Yan. Investigating messaging protocols for the internet of things (iot). *IEEE Access*, 8:94880–94911, 2020.

[4] Mohammad Almseidin, Maen Alzubi, Szilveszter Kovacs, and Mouhammd Alkasassbeh. Evaluation of machine learning algorithms for intrusion detection system. In *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, pages 000277–000282, 2017.

[5] Oluleye H Babatunde, Leisa Armstrong, Jinsong Leng, and Dean Diepeveen. A genetic algorithm-based feature selection. 2014.

[6] M Dachyar, Teuku Yuri M Zagloel, and L Ranjaliba Saragih. Knowledge growth and development: internet of things (iot) research, 2006–2018. *Heliyon*, 5(8), 2019.

[7] Datta H Deshmukh, Tushar Ghorpade, and Puja Padiya. 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)*, pages 1–7. IEEE, 2014.

[8] Dario Di Dario, Valeria Pontillo, Stefano Lambiase, Filomena Ferrucci, and Fabio Palomba. Security testing in the wild: An interview study. In *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 191–198. IEEE, 2023.

[9] Gordon Fraser and Andrea Arcuri. Whole test suite generation. *IEEE Transactions on Software Engineering*, 39(2):276–291, 2012.

[10] Giammaria Giordano, Fabio Palomba, and Filomena Ferrucci. On the use of artificial intelligence to deal with privacy in iot systems: A systematic literature review. *Journal of Systems and Software*, 193:111475, 2022.

[11] Mark Harman, S Afshin Mansouri, and Yuanyuan Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys (CSUR)*, 45(1):1–61, 2012.

[12] Gunes Kayacık, A. Zincir-Heywood, and Malcolm Heywood. Selecting features for intrusion detection: A feature relevance analysis on kdd 99. 01 2005.

[13] Dong Seong Kim, Ha-Nam Nguyen, and Jong Sou Park. Genetic algorithm to improve svm based network intrusion detection system. pages 155–158 vol.2.

[14] Yuan-Cheng Li and Zhong-Qiang Wang. An intrusion detection method based on svm and kpca. In *2007 International Conference on Wavelet Analysis and Pattern Recognition*, volume 4, pages 1462–1466, 2007.

[15] Tao Lu, Youpeng Huang, Wen Zhao, and Jie Zhang. 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)*, pages 370–374. IEEE, 2019.

[16] Amir Makhshari and Ali Mesbah. Iot bugs and development challenges. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 460–472. IEEE, 2021.

[17] Phil McMinn. Search-based software test data generation: a survey. *Software testing, Verification and reliability*, 14(2):105–156, 2004.

[18] Francesca Meneghello, Matteo Calore, Daniel Zucchetto, Michele Polese, and Andrea Zanella. Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices. *IEEE Internet of Things Journal*, 6(5):8182–8201, 2019.

[19] Melanie Mitchell. Genetic algorithms: An overview. In *Complex.*, volume 1, pages 31–39. Citeseer, 1995.

[20] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[21] Eddy Prasetyo Nugroho, Taufik Djatna, Imas Sukaesih Sitanggang, Agus Buono, and Irman Hermadi. A review of intrusion detection system in iot with machine learning approach: Current and future research. In *2020 6th International Conference on Science in Information Technology (ICSITech)*, pages 138–143, 2020.

[22] Ibrahim Obeidat, Nabhan Hamadneh, Mouhammd Alkasassbeh, Mohammad Almseidin, and Mazen AlZubi. Intensive pre-processing of kdd cup 99 for network intrusion classification using machine learning techniques. 2019.

[23] omitted for double anonymous review. Web Appendix of the paper. https://figshare.com/s/f7fc3d84dac00410bab3. Online.

[24] Patrick M. Reed, Barbara S. Minsker, and David E. Goldberg. *The Practitioner's Role in Competent Search and Optimization Using Genetic Algorithms*, pages 1–9.

[25] Suchet Sapre, Pouyan Ahmadi, and Khondkar Islam. 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.

[26] Chayanika Sharma, Sangeeta Sabharwal, and Ritu Sibal. A survey on software testing techniques using genetic algorithm. *arXiv preprint arXiv:1411.1154*, 2014.

[27] Mohammad Khubeb Siddiqui and Shams Naahid. Analysis of kdd cup 99 dataset using clustering based data mining. *International Journal of Database Theory and Application*, 6(5):23–34, 2013.

[28] Gary Stein, Bing Chen, Annie S. Wu, and Kien A. Hua. Decision tree classifier for network intrusion detection with ga-based feature selection. In *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2*, ACM-SE 43, page 136–141, New York, NY, USA, 2005. Association for Computing Machinery.

[29] Ravindra Gupta Swati Paliwal. Denial-of-service, probing & remote to user (r2l) attack detection using genetic algorithm. *International Journal of Computer Applications*, 60(19):57–62, December 2012.

[30] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009.

[31] John R Vacca. *Computer and information security handbook*. Newnes, 2012.

[32] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[33] Bruno Bogaz Zarpelão, Rodrigo Sanches Miani, Cláudio Toshio Kawakani, and Sean Carlisto de Alvarenga. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, 84:25–37, 2017.