2021-07-01

Grammatical Evolution for Detecting Cyberattacks in Internet of Things Environments

Hasanen Alyasiri
University of Kufa, Iraq

John Clark
The University of Sheffield, UK, john.clark@sheffield.ac.uk

Ali Malik
Technological University Dublin, ali.malik@tudublin.ie

See next page for additional authors

Funder: Science Foundation Ireland (SFI)

Grammatical Evolution for Detecting Cyberattacks in Internet of Things Environments

Hasanen Alyasiri *, John A Clark †, Ali Malik ‡, Ruairí de Fréin ‡
*Department of Computer Science, University of Kufa, Iraq
hasanen.alyasiri@uokufa.edu.iq
†Department of Computer Science, The University of Sheffield, United Kingdom
john.clark@sheffield.ac.uk
‡School of Electrical and Electronic Engineering, Technological University Dublin, Ireland
ali.malik, ruairi.defrein@tudublin.ie

***Abstract*—The Internet of Things (IoT) is revolutionising nearly every aspect of modern life, playing an ever greater role in both industrial and domestic sectors. The increasing frequency of cyber-incidents is a consequence of the pervasiveness of IoT. Threats are becoming more sophisticated, with attackers using new attacks or modifying existing ones. Security teams must deal with a diverse and complex threat landscape that is constantly evolving. Traditional security solutions cannot protect such systems adequately and so researchers have begun to use Machine Learning algorithms to discover effective defence systems. In this paper, we investigate how one approach from the domain of evolutionary computation - grammatical evolution - can be used to identify cyberattacks in IoT environments. The experiments were conducted on up-to-date datasets and compared with state-of-the-art algorithms. The potential application of evolutionary computation-based approaches to detect unknown attacks is also examined and discussed.**

***Index Terms*—Grammatical Evolution, IoT Security, Unknown Attacks**

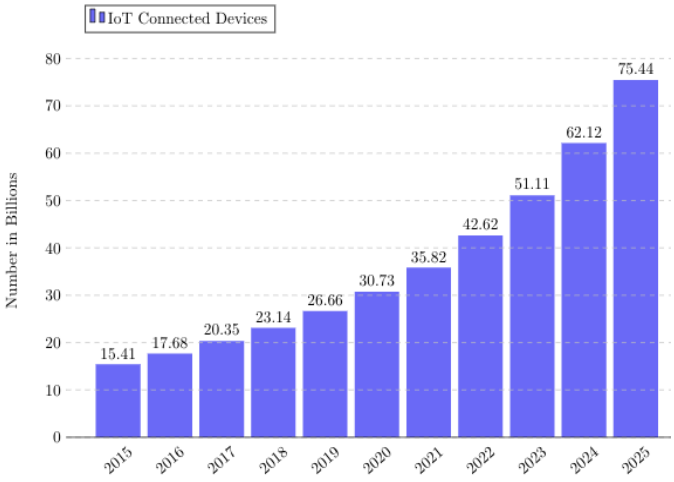

Fig. 1. IoT: Number of devices worldwide from 2015 to 2025 [5].

I. Introduction

The Internet of Things (IoT) is composed of many devices whose augmentation with processors and network cards provide connection to the Internet and allow the exchange of data. Things, in this context, refer to various devices ranging from complex cloud servers to simple sensors that can be managed by users using the web, app or other types of interfaces [1]. These devices can interact with each other with minimum human involvement. IoT technologies help improve life quality by introducing smart applications in education, healthcare, cities, factory, and many more domains [2] [3] [1]. IoT is one of the fastest-growing industries, despite its fairly recent initiation (between 2008 and 2009) [4]. According to the survey conducted by Statista [5], the count of IoT connected devices will be more than 50 billion by 2023 and 75 billion by 2025, as presented in Figure 1. Despite this growth, experts have recognised that many IoT devices are still inherently vulnerable and a weak link in security [6].

Recently, Mirai malware was used to compromise various vulnerable IoT devices, namely cameras, modems, and routers, and turn them into botnets [6]. These botnets were then used, in September 2016, to launch a large scale Distributed Denial of Service (DDoS) attack against the Internet infrastructure firm Dyn, which paralysed access to several high-profile websites. This showed how dangerous the consequences are if IoT systems are compromised. Not only are the IoT users affected, but a whole range of systems such as websites, applications, social networks and servers [7]. In addition, the large-scale and complex nature of IoT paradigms has increased networks vulnerability [6]. Current traditional security solutions, such as encryption, authentication, and access control, though important, are inadequate for addressing complex security requirements of IoT systems [1] [7]. Traditional approaches must be augmented with new methods to provide adequate protection for IoT systems. Over the past few years, and of specific interest to the work reported in this paper, the field of Machine Learning (ML) has advanced considerably for solving various real-world problems. For this reason, many researchers have begun to deploy ML algorithms to improve the security of various systems including IoT [1] [7]. Our work explores a particular approach as indicated below.

Evolutionary Computation (EC) algorithms, a subset of ML algorithms, have successfully produced intelligent solutions for different fields, for instance, computational biology, medical science, finance, engineering and many others [2] [8].

Additionally, the study in [2] has concluded that EC techniques have the potential to identify cybersecurity risks including those of IoT systems. Furthermore, solutions evolved using EC techniques possess a number of attractive characteristics such as being readable and lightweight, and employing fewer features than other ML algorithms [9].

In this paper, we investigate the use of one EC algorithm, Grammatical Evolution (GE), to evolve rules for detecting attacks against IoT environments. There are various studies that have adopted GE algorithm to address security problems in different fields including mobile ad hoc networks [10], traditional networks [11], phishing threats [12], and botnet detection [13]. However, this paper is the first study to use GE algorithm for the purpose of protecting IoT environments. Moreover, a recent study [7] has shown that *unknown* attacks are an ongoing issue for IoT systems. Hence, we will evaluate how well GE algorithm can address this issue.

This paper is organised as follows: Section II summarises background and related work; Section III explains the methodology; Section IV describes the experiments conducted and the results obtained; and Section V concludes.

II. BACKGROUND

IoT systems have become an active research area in cybersecurity. This is due to the wide-spread adoption of IoT devices and inadequacy of currently used protection techniques [14]. IoT presents a significantly increased threat landscape: a great many traditional server and networking threats are present but the proliferation of end devices, with their varying capabilities and resources, offers many new points to attack [15].

Turning IoT devices into a botnet is one particular threat. Botnets refer to compromised devices that are controlled by an attacker using a special command and control channel [13]. These botnets can be used to perform a number of malicious activities: DDoS, keylogging, spamming, phishing and others [13] [15]. Another security risk for IoT environments comes from its communication protocol, such as Message Queuing Telemetry Transport (MQTT) [14]. MQTT is a message transport protocol used for machine-to-machine communications (the basis for IoT). It has a number of attractive features including being lightweight, having low bandwidth requirements and being highly effective [3]. These features make it the prominent protocol for IoT systems. However, such a protocol is vulnerable and attackers could exploit this vulnerability to launch attacks such as aggressive scans and brute-force attacks [3] [14]. Both botnet-based and MQTT-based attacks are addressed in this study.

In the literature, several researchers have built ML-based security methods for IoT systems, as outlined below:

Aydogan et al. [16] proposed using genetic programming to detect attacks in the context of industrial IoT applications. Two types of attacks were investigated, namely Hello flood attacks and Version number attacks, which target the routing protocol of low-power and lossy IoT networks. The proposed system achieved very satisfying outcomes (high true positive and low false positive rates).

Koroniotis et al. [15] used Machine and Deep Learning algorithms to detect threats to IoT frameworks. These algorithms were: Support Vector Machines (SVMs), Recurrent Neural Networks (RNNs) and Long-Short Term Memory Recurrent Neural Network (LSTM-RNNs). Different attack scenarios were considered including probing attacks, Denial of Service (DoS) attacks and information theft. The results showed these algorithms were capable of detecting such attacks with a detection rate of 100% in the best-case scenario.

Aldhaheri et al. [17] developed a hybrid detection system consisting of Deep Learning and Dendritic Cell Algorithm (DeepDCA) for detecting attacks against IoT frameworks. Threats, such as DoS, DDoS, Information gathering and theft, were investigated. Before training the classifier, they applied the Information Gain approach to select the most effective (i.e. informative) features from the search space to minimise the computational cost and enhance accuracy. DeepDCA demonstrated better performance than other ML algorithms including SVM, Naive Bayes (NB), K Nearest Neighbor (KNN) and Multilayer Perceptron (MLP).

Hindy et al. investigated the use of six different ML algorithms to detect MQTT-based attacks [14]. These attacks were aggressive scans, User Datagram Protocol (UDP) scans, Sparta SSH brute-force attacks, and MQTT brute-force attacks. Three sets of features representing behaviours in the IoT environment were extracted, from packet-based data and two types of flow-based (i.e. uni-directional and bi-directional) data. Their work demonstrated the effectiveness of detecting such attacks using ML algorithms and bi-directional flow-based features.

III. PROPOSED METHODOLOGY

In this paper, our objective is to use the Grammatical Evolution (GE) algorithm to evolve programs to detect cyberattacks in IoT environments. GE is a population-based optimisation algorithm inspired largely by the mechanism of Darwinian evolution. It seeks to evolve ever-fitter populations of candidate solutions for the intended environment [18]. It was devised by Ryan and O'Neill in 1998 [19].

GE is capable of automatically evolving complete programs in any language. It typically uses sequences of non-negative integers of variable length to represent individuals. The grammars used to express the space of programs are specified in Backus-Naur Form (BNF), a notation for specifying context-free grammars. The grammar is denoted by a four-tuple $\{N,\ T,\ P,\ S\}$, where N is a non-terminal set, T is the terminal set (i.e. inputs), P is the production rules set, and S is the start symbol of the grammar from which the generation process begins. Sentences in the language are concrete expressions containing only terminals. They are derived by successive application of production rules to non-terminals (i.e. replacing a non-terminal that appears on the left-hand side of a rule with one of its productions on the right-hand side). A typical approach is to expand the leftmost non-terminal. A candidate solution is a non-negative integer sequence. The elements of such a sequence are successively interpreted as indices into the relevant production rules (i.e.

they indicate which rules to apply). If a non-terminal can be expanded via k rules (indexed by 0 to $(k-1)$), then an integer value V is deemed to indicate the $(V mod\ k)th$ rule. Consider the following grammar as an example:

N = {expr, op, pre-op, var}
T = { +, −, /, ∗, sqrt, sin, cos, tan, x, 1.0}
S = <expr>

and P can be represented as

<expr> ::= <expr><op><expr> 0
| <pre-op><expr> 1
| <var> 2
<op> ::= + 0
| − 1
| / 2
| ∗ 3
<pre-op> ::= sqrt 0
| sin 1
| cos 2
| tan 3
<var> ::= x 0
| 1.0 1

The GE genomes are expressed as variable-length strings of 8-bit elements. Each set of 8 bits forms what generally is referred to as a *codon value* which is used to select the production rule from the BNF grammar. Consider the following genome of a GE individual represented by a sequence of integers, for instance $\{210,\ 35,\ 46,\ 67,\ 136,\ 53,\ 143,\ 25\}$. Since S is the start symbol. There is only one production rule that starts with S (mapping S to $< expr >$) and so we start the decoding process by substituting S with $< expr >$. $< expr >$ has 3 rules and so we interpret 210 as 210 mod 3, i.e. as rule 0, and so we expand $< expr >$ as $< expr >< op >< expr >$. We now expand the leftmost non-terminal in this expression. Once again, $< expr >$ has 3 possible rules to expand it, and we select rule (35 mod 3), i.e. rule 2. Applying this rule gives the string $< var >< op >< expr >$. This continues until non non-terminals symbols remain. The final expression evolved given the genome and the grammar is '$x * sin(1.0)$'. If we run out of integers before a string is produced comprising entirely terminal elements, a recovery strategy needs to be adopted, e.g. continuing from the first element of the array, a strategy called *wrapping* [19]. However, if a valid sentence (i.e. comprising only terminals) is not produced within a certain number of wraps, GE will consider this chromosome to be invalid.

GE creates the initial population from randomly produced chromosomes, each considered as a candidate solution to the problem [20]. Generation by generation, GE applies genetic operators (i.e. crossover and mutation) to produce new solutions, hopefully better, that replace the old ones. The single-point crossover is used where two positions are chosen on each codon sequence at random and their genetic contents are swapped beginning from these positions. The mutation operator mutates at random on each bit in a GE genome with a predetermined mutation probability. The mutation is essential to preserve the diversity of GE individuals through the evolutionary process [20]. This will continue until the termination condition is met (i.e. an ideal solution evolved or the maximum number of generations reached). The general steps of GE algorithm are stated in Algorithm 1.

Algorithm 1: The general steps of GE

```
Create initial random population;
while termination conditions not met do
    Evaluate the fitness of each individual;
    Apply genetic operators to the individuals;
    Create a new population;
end
Return the best individual;
```

The steady-state replacement mechanism is applied in the GE algorithm by default to ensure the validity of solutions in evolution [18]. Invalid solutions are given the lowest fitness value, however, using a simple replacement mechanism these solutions may possibly remain in future generations. These invalid solutions may cause a delay in the evolution process. Thus, the positive effect of having the steady-state mechanism is its capability to replace the worst solutions in a population with new solutions.

A. Fitness Function

The fitness function measures how well an individual solves the problem at hand. In our experiments, the fitness value is calculated using the Matthews Correlation Coefficient (MCC). A new study [21] has shown that MCC provided a more informative and truthful score in assessing binary classifiers than conventional metrics, especially when an unbalanced dataset is investigated. MCC relies on the four categories of the confusion matrix (i.e. true positives, true negatives, false positives, and false negatives). The following formula is used to calculate the MCC score:

$$MCC = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)*(FN+TN)*(FP+TN)*(TP+FN)}} \quad (1)$$

True Positives (TP) is the count of attack instances actually identified as attacks. True Negatives (TN) is the count of normal events identified correctly as normal. False Positives (FP) is the count of normal events identified as attacks. False Negatives (FN) is the count of actual attack instances incorrectly identified as normal events. The MCC score produces values in the range of -1 to 1. $MCC = -1$ indicates a completely incorrect classification while $MCC = 1$ shows a completely correct classification. The fitness value is given by:

$$f = 1 - MCC \quad (2)$$

The fitness value will be equal 0 when all instances identified correctly (i.e. the ideal).

B. GE Grammars and Settings

To evolve detection rules, GE uses a BNF grammar that produces 'if' statements, which raise an alarm when the condition is true. Function nodes consist of a set of mathematical, relational, and logical operators. These operators are of two types: a binary which takes 2 inputs and a unary which takes one input only. Relational nodes take any 2 inputs and return an output of a Boolean type whereas logical operators take 2 Boolean inputs and return an output of a Boolean type. The output node takes relational or logical operators only. The BNF grammar of IoT attacks detection problem is provided below:

<rule> ::= if (<condition>) {raise alarm ( )}
<condition> ::= <condition>AND<condition>
| <condition>OR<condition>
| <rel.op>
<rel.op> ::= <op>(<expr>, <expr>)
<op> ::= >, >=, <, <=
<expr> ::= <binary.op>(<expr> , <expr>)
| <unary.op>(<expr>)
| <var>
<unary.op> ::= sqrt | abs | ceil | exp | floor | log | sin | cos
| tan | tanh
<binary.op> ::= + | − | ∗ | protected (/) | power | max | min
<var> ::= The feature set given by used datasets

GE was implemented using an R package named gramEvol [20] in this study. GE settings used in the experiments had a population size of 1000, the maximum count of generations was set to 50, and without elitism. The rest of the settings is automatically determined by the package. The maximum search depth in the case of cyclic grammar and is limited to the number of production rules in the grammar.

IV. Experimental Design and Evaluation

The section includes two types of experiments: the evolution of detection rules for IoT cyberattacks and for unknown attacks.

A. Description of the Benchmark Datasets

To demonstrate the effectiveness of the proposed method we used two recently contributed datasets: BoT-IoT [15] and IoT-MQTT [22]. The BoT-IoT dataset was generated in the Research Cyber Range lab of UNSW Canberra. This dataset represents a realistic IoT environment and has many attractive features, for instance a realistic testbed configuration, a realistic traffic, diverse attack scenarios, etc, compared to other datasets. The attacks traffic in this dataset are described as follows [15]:

Probing attacks are malicious activities in which an attacker collects information about vulnerabilities in the targeted system using scanning software, for instance, port scanning and operating system fingerprinting. This collected information can help attackers evade system security controls.

Denial of Service is an intrusion action where an attacker attempts to disrupt service(s), and thus deny legitimate users access. Usually, compromised machines are used to launch such kind of attacks. In this dataset, two types of Denial of Service are performed, namely DoS and DDoS, which differ in the volume of the attack. Moreover, they launched both DoS and DDoS using various network protocols.

Information theft is an attempt to compromise the security of a machine in order to steal sensitive information. They implement two such attacks, namely data theft and keylogging. In data theft, the attacker downloads the sensitive data from a targeted machine after compromising it. In keylogging, the attacker records the user activities (i.e. keystrokes) to help leak sensitive credentials.

We used the original split of the training and testing portions (see Table I). The Argus tool was used to extract the feature set that describes various behaviours in the IoT environment. In addition, they created new features based on the statistics of groups of network flows in which models of different attacks can be identified. In the original dataset, a collection of the 10 best features has been given, which we used in our implementations (see Table II).

TABLE I
BoT-IoT Dataset Distribution

Category	Training	Testing
DDoS	1,541,315	385,309
DoS	1,320,148	330,112
Reconnaissance	72,919	18,163
Information Theft	65	14
Normal	370	107

TABLE II
BoT-IoT best 10 features Description [15]

Feature Name	Description
seq	Argus sequence number
stddev	Standard deviation of aggregated records
N_IN_Conn_P_SrcIP	No. of inbound connections per source IP
N_IN_Conn_P_DstIP	No. of inbound connections per destination IP
state_number	Numerical representation of feature state
mean	Average duration of aggregated records
drate	Destination-to-source packets per second
srate	Source-to-destination packets per second
min	Minimum duration of aggregated records
max	Maximum duration of aggregated records

In the IoT-MQTT dataset, there are two types of attacks, namely scanning and brute-force. In a scanning attack, an attacker scans the connected devices to a network to gather valuable information about services and operating systems they are running. This information can be used to launch attacks. They implemented two types of this attack namely aggressive scan and UDP scan. In a brute-force attack an intruder tries all possible combinations to guess critical information such as login, encryption keys, etc. Sparta SSH brute-force and MQTT brute-force attacks were performed.

The tcpdump tool was used to collect Ethernet traffic from the simulated environment. Packet-based and flow-based

features were generated, which their experiments showed that flow-based types were better in discriminating between normal and attack behaviour [14]. The two types of flow-based feature (i.e. unidirectional and bidirectional) were used in our experiments (see Table III). In the case of bidirectional flows, each feature has two values: one represents the forward flow and the other represents backward flow.

TABLE III
IOT-MQTT USED FEATURES DESCRIPTION [22]

Feature Name	Description
num_pkts	Number of Packets in the flow
mean_iat	Average inter arrival time
std_iat	Standard deviation of inter arrival time
min_iat	Minimum inter arrival time
max_iat	Maximum inter arrival time
mean_pkt_len	Average packet length
num_bytes	Number of bytes
num_psh_flags	Number of push flag
num_rst_flags	Number of reset flag
num_urg_flags	Number of urgent flag
std_pkt_len	Standard deviation packet length
min_pkt_len	Minimum packet length
max_pkt_len	Maximum packet length

We randomly partitioned this dataset into 70% for the training and 30% for the testing. Each set of the unidirectional data contained around 24% attacks whereas the bidirectional data contained around 27% attacks in each set.

B. Evaluation Metrics

The metrics utilised to evaluate GE algorithm performance are: Detection Rate ($DR = \frac{TP}{(TP+FN)}$) which provides the fraction of attack instances that are identified. Accuracy ($\frac{(TP+TN)}{(TP+FP+TN+FN)}$) indicates of how well all classes (i.e. attacks and normal) are identified. In addition, we calculate the percentage of misclassified instances using Error Rate ($ERR = \frac{(FP+FN)}{(TP+FP+TN+FN)}$). Finally, the MCC score is reported.

C. Results

To remove dependence on the initial random seed from the results, we report the average plus the Standard Error (SE) of the average for 20 independent runs. The SE gives the dispersion in the average with several experiments been conducted. It measured using ($\sigma/\sqrt{N}$) where N in our case indicates the count of runs. The change in the fitness value (i.e. MCC score) of the best-evolved rule throughout the generations is shown in Figure 2. The best rule from the IoT-MQTT experiment appeared to display performance on the test set as effective as on the training set.

As seen from the results in Table IV, GE algorithm showed an excellent performance in terms of DR, ERR and Accuracy in the BoT-IoT dataset. GE algorithm achieved similar and in some cases better results than other ML algorithm outcomes reported in the literature. The performance of LSTM [15] tested on BoT-IoT dataset were 99.75% DR and 99.74% Accuracy, while DeepDCA [17] gave 98.36% DR and 98.73% Accuracy. However, the MCC score was low. An examination

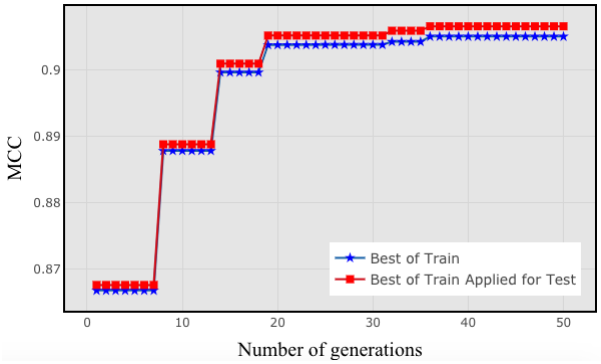

Fig. 2. The MCC score of the best rule in different generations.

of the outcomes shows that evolved rules were classifying the normal behaviour with a range from 26.18% to 58.87%. This is due to the massively unbalanced BoT-IoT dataset where normal instances represent 0.01%. Hence, balancing techniques could have yielded different outcomes which we will examine in the future.

In the IoT-MQTT dataset, GE algorithm accomplished an excellent performance in terms of DR rate using both types of flow features. Overall, the bidirectional features showed better performance than unidirectional ones. In comparison, various ML algorithms [14] achieved DR range from 78% to 99.98% and from 96.61% to 99.97% for unidirectional and bidirectional flow features, respectively.

TABLE IV
GE ALGORITHM PERFORMANCE (%)

	BoT-IoT	IoT-MQTT (Uni)	IoT-MQTT (Bi)
DR	99.99 ± 0	99.41 ± 0.05	99.20 ± 0.13
ERR	0.01 ± 0	3.29 ± 0.15	2.05 ± 0.22
Accuracy	99.98 ± 0	96.70 ± 0.15	97.94 ± 0.22
MCC	46.96 ± 1.08	91.65 ± 0.37	95.06 ± 0.52

The following is one of the best evolved rules produced by GE algorithm taken from the BoT-IoT experiment:

if (N_IN_Conn_P_SrcIP $<$ exp(ceil(N_IN_Conn_P_DstIP)) + power (max, log(state_number))), **raise alarm ( )**

As shown in the rule, GE algorithm used 4 out of 10 features available during the evolution process. In addition, the readable output (a particular strength of GE) can give useful insights into how the algorithm solves the specified problem.

D. GE algorithm for Detecting Unknown Attacks

Many IoT protection systems have been produced using signature-based or normal behaviour specifications. However, these are inadequate for detecting unknown/zero-day attacks [7]. We assessed the suitability of GE to learn characteristics of known attacks which are often mutations of previous attacks. The BoT-IoT dataset was adopted for this experiment. There

are 4 kinds of attacks in this dataset (see Table I). We have ensured that training and testing sets contain different attacks. To do so, for each attack kind that mimics an unknown attack we removed its training samples and added them to the testing portion. For each investigation, we ran the algorithm 20 times which makes the total count of the independent runs for all attacks is (20 * 4 = 80). Fig. 3 shows the spread of responses regarding the detection rate for each unknown attack.

Fig. 3. Boxplots show the distribution of each unknown attack detected by GE algorithm (each boxplot represents an experiment of 20 independent runs).

The boxplots indicate that GE algorithm showed high and steady performance in detecting unknown attacks, specifically DDoS, DoS, and Information theft. However, it has achieved the lowest DR for Reconnaissance attacks in comparison to other unknown attacks. To the best of our knowledge, this is due to the nature of this attack, which may target the port. Consequently, for the future work, we need to include other types of feature such as port statistics. In general, these results suggest that GE algorithm has successfully classified unknown attacks that the system was not trained on.

V. Conclusion

IoT devices and services have become a favourite target of cybercriminals. In this paper, we presented a preliminary investigation of the GE algorithm to detect cyberattacks against IoT environments. Our proposed method is capable of mapping the input vector space (i.e. extracted features) into a decision space to discriminate between classes (i.e. normal vs. attack). In addition, the experimental results demonstrated that the GE algorithm showed a robust performance in detecting attacks that were absent from the training stage. In future, we plan to use GE to synthesise an ensemble model (i.e. a collection of detectors) to detect attacks on IoT environments.

ACKNOWLEDGEMENT

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under the Grant Number 15/SIRG/3459.

References

[1] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (iot) security," *IEEE Communications Surveys & Tutorials*, 2020.

[2] H. He, C. Maple, T. Watson, A. Tiwari, J. Mehnen, Y. Jin, and B. Gabrys, "The security challenges in the iot enabled cyber-physical systems and opportunities for evolutionary computing & other computational intelligence," in *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 1015–1021.

[3] M. Harsha, B. Bhavani, and K. Kundhavai, "Analysis of vulnerabilities in mqtt security using shodan api and implementation of its countermeasures via authentication and acls," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2018, pp. 2244–2250.

[4] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[5] S. R. Department, "Internet of Things - number of connected devices worldwide 2015-2025," 2016, (Accessed: 2021, Feb 2). [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[6] Security Report 2020 — Check Point Software. (2020, June 26). [Online]. Available: https://pages.checkpoint.com/cyber-security-report-2020.html

[7] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[8] R. de Fréin and L. O'Farrell, "Distance-based cluster head election for mobile sensing," in *2018 12th International Conference on Sensing Technology (ICST)*. IEEE, 2018, pp. 196–201.

[9] S. Sen, "A survey of intrusion detection systems using evolutionary computation," in *Bio-inspired computation in telecommunications*. Elsevier, 2015, pp. 73–94.

[10] S. Şen and J. A. Clark, "Evolving intrusion detection rules on mobile ad hoc networks," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2008, pp. 1053–1058.

[11] D. Wilson and D. Kaur, "Using grammatical evolution for evolving intrusion detection rules," *WSEAS Transactions on Systems*, vol. 6, no. 2, pp. 346–351, 2007.

[12] H. Alyasiri, J. A. Clark, and D. Kudenko, "Evolutionary computation algorithms for detecting known and unknown attacks," in *International Conference on Security for Information Technology and Communications*. Springer, 2018, pp. 170–184.

[13] S. Yilmaz and S. Sen, "Early detection of botnet activities using grammatical evolution," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2019, pp. 395–404.

[14] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine learning based iot intrusion detection system: An mqtt case study," *arXiv preprint arXiv:2006.15340*, 2020.

[15] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[16] E. Aydogan, S. Yilmaz, S. Sen, I. Butun, S. Forsström, and M. Gidlund, "A central intrusion detection system for rpl-based industrial internet of things," in *2019 15th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 2019, pp. 1–5.

[17] S. Aldhaheri, D. Alghazzawi, L. Cheng, B. Alzahrani, and A. Al-Barakati, "Deepdca: Novel network-based detection of iot attacks using artificial immune system," *Applied Sciences*, vol. 10, no. 6, p. 1909, 2020.

[18] M. O'Neill and C. Ryan, "Grammatical evolution," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 349–358, 2001.

[19] C. Ryan, J. J. Collins, and M. O. Neill, "Grammatical evolution: Evolving programs for an arbitrary language," in *European Conference on Genetic Programming*. Springer, 1998, pp. 83–96.

[20] F. Noorian, A. M. de Silva, and P. H. Leong, "gramevol: Grammatical evolution in r," *Journal of Statistical Software*, vol. 71, no. i01, 2016.

[21] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, p. 6, 2020.

[22] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens, "Mqtt internet of things intrusion detection dataset," 2020. [Online]. Available: http://dx.doi.org/10.21227/bhxy-ep04