IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 30, 2021, accepted April 14, 2021, date of publication April 16, 2021, date of current version April 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3073785

Anomaly Detection in IoT Networks: From Architectures to Machine Learning Transparency

ALEKS HUČ AND DENIS TRČEK
Faculty of Computer and Information Science, University of Ljubljana, 1000 Ljubljana, Slovenia
Corresponding author: Denis Trček (denis.trcek@fri.uni-lj.si)

This work was supported by the Slovene Research Agency ARRS through the Research Program Pervasive Computing under Grant P2-0359.

ABSTRACT Machine learning (ML) is becoming an integral part of networks security arsenal, where Internet of Things (IoT) structures play an increasingly important role. However, IoT networks have many specific requirements, mostly due to limited energy availability and stringent computing resources. This results in limitations for traditional ML approaches to security, in particular for anomaly detection. Consequently, new focuses for solutions that range from architectural to data processing ones are necessary. Therefore, appropriate lightweight ML algorithms have to be designed and deployed in appropriate architectural settings, which is the main contribution of this paper. In addition, insights into ML functioning are needed to better understand the observed anomalies. To enable these insights (and support a wider applicability of ML based approaches), the results have to be as explainable as possible. The research presented in this paper addresses this problem through the functional and data transparency of ML applications, tailored to the specifics of anomaly detection in IoT networks. To tackle accordingly also the architectural issues, the presented approach builds on the well-established layering principle from computer communications reference models. This principle not only supports flexibility but also increases security in these new environments of growing importance.

INDEX TERMS Computer networks, Internet of Things, security architectures, anomaly detection, machine learning, functional transparency, data transparency.

I. INTRODUCTION

Internet of Things (IoT) devices are penetrating contemporary networks at a still surprising rate. These devices along with their networks (and various IoT-based agglomerations) are about to become the mostly numerous and dominant kind of devices in the global network [1]. They need increased attention because of their importance to digital transformation, which is currently extending from services to tangible goods sectors. Consequently, digital transformation also affects critical infrastructures deployments and deployments in industrial production settings, where one central structure are cyber-physical systems (CPS). These systems are largely built on top of IoT [2]. Therefore, security provisioning for IoT structures is becoming a must and anomaly detection is one of the top priorities.

The associate editor coordinating the review of this manuscript and approving it for publication was Theofanis P. Raptis.

IoT devices have many specific characteristics, but the most commonly mentioned ones are limited computing resources and constrained energy availability. Related to these, but almost unaddressed, are architectural deployment consequences. In addition, ML specifics related to their transparency are important if one wants the results to be explainable and enable further advances in the field [3]. Therefore, this paper presents a new architectural approach to lightweight ML based anomaly detection deployments tailored to IoT networks, being further complemented by transparency of ML functioning. More precisely, explainability of ML results remains an important topic not only in the ML domain, but also in the areas of ML deployments like security. This requires the selection of an appropriate family of ML algorithms, which we undertake in this research paper.

The paper is structured as follows. The necessary basics, which are crucial for ML based anomaly detection in IoT networks, are given in Section II. After identifying security relevant issues, a new architectural approach and a new

algorithm are given in Section III. ML based transparency issues and proposed solutions are covered next in the same section. In Section IV there is a discussion, followed by conclusions in Section V, with references and the authors' CVs at the end.

II. OVERVIEW OF THE FIELD AND IDENTIFYING CHALLENGES

To provide an overview of the filed, anomaly detection in traditional networks will be addressed in this section first, followed by ML related issues.

Traditional networks follow a classical TCP/IP architectural model [4]. When aligned with the OSI reference model [5], this model consists of physical layer that enables physical connection of devices to a medium, a link layer that meets particular medium requirements, a network layer that basically covers routing with an IP protocol, and a transport layer that provides end-to-end connectivity with TCP and UDP protocols. It ends with an application layer (no explicit presence of presentation or session layer), where services like WWW and e-mail are offered.

Due to computational power and energy resources related constraints in IoT networking, TCP/IP architectural model has been adapted and this adapted model is shown in Fig.1.

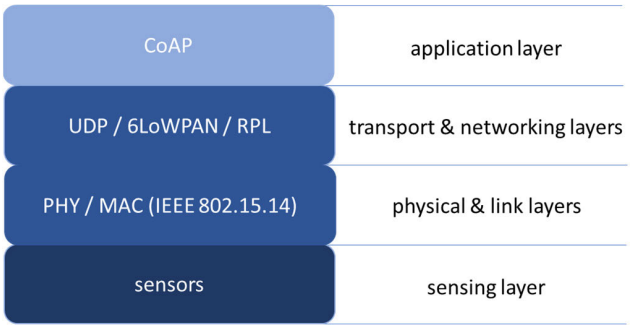


FIGURE 1. IoT network model with most common protocols.

The IoT model has a sensing layer that is often specific and covers physical and link layers. The network layer is covered by 6LoWPAN [6], which is an IPv6 adapted protocol, while routing is provided by an adapted energy efficient RPL protocol [7]. The transport connection is often covered by a traditional UDP protocol. The application protocol is most often CoAP [8], which is similar to HTTP [4] but has lower latency and draws less power from the device that it runs on.

Putting this reference model into a typical IoT architectural setting, Fig. 2 is obtained. The weakest building elements of IoT networks are sensors and actuators. To enable their further integration via physical and link connections, hubs are deployed, which are able to run the IoT stack protocols. Hubs communicate with one another and enter the global internet (or cloud) though one (or more) edge routers, which may also convert IPv6 to IPv4.

From a security point of view the key anomaly-based detection points for protecting IoT network are hubs

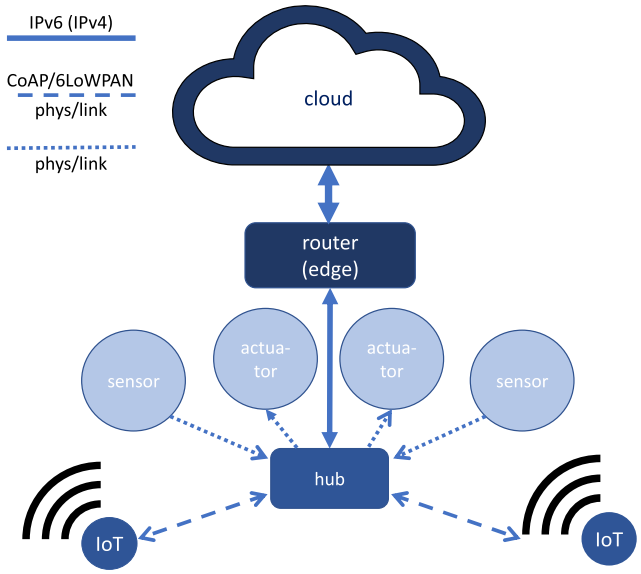


FIGURE 2. Deployment architecture of IoT networks and their integration with the internet.

(computationally relatively weak), and edge routers (relatively strong with sufficient computing resources).

A. ANOMALY DETECTION

Based on the brief background provided above, let us now look at anomaly detection in traditional networks (for an extensive overview of this area the reader is advised to look at [9]). As stated in [9], monitoring network traffic is essential to preserving network security, with anomaly detection as one key goal of this monitoring. In classical settings monitoring is done on hosts with installed intrusion detection systems (IDSs). But such host-based approaches cause scalability issues, have limited access to data (i.e. they provide no "big picture" of the situation) and can run only on appropriate, and sufficiently powerful hosts. Therefore, network-based approaches are often deployed instead. In this case, the so-called probes are located at central observation point routers, which are located at, for example, endpoints of trunk connection lines. Here they are capturing packets at the network layer and inspecting their payloads. But this traditional approach is becoming ineffective as the packets' payloads are often encrypted nowadays, and there are processing demands for duplication and storage of whole packets with ever-increasing traffic.

Therefore, storing only IP specific packet data (IP addresses and parameters like time stamps, the number of packets exchanged and flags) in the form of streams is becoming a frequently deployed option. Streams-based capturing is also supported by the IETF standard IPFIX [10]. However, one should be cautious – streams-based anomaly detection leads to lower granularity compared to packets capturing and may introduce artefacts that can make it less efficient [11]. In general, network-based IDSs (compared to host-based IDSs) are weaker when it comes to compromise detection and have higher false positives. So, due to the nature of captured

data they warn about the presence of attacks regardless of their success, i.e. if attacks have led to the compromise of a device. False positives can also be the result of data transformations (from raw packets to streams).

Finally, and more specifically related to the research presented in this paper, the work given in [12] proves that network flow data analysis can achieve good performance. It is important to add that this research is focused on the deployment of unsupervised clustering algorithm(s).

B. ML TRANSPARENCY ISSUES

Despite the huge recent success of ML solutions, one fundamental open problem that is still a matter of intensive research is explainability, also referred to as transparency. The issue is particularly acute for more complex ML solutions like deep neural networks (DNN). In short, an ML implementation successfully learns how to solve a problem (categorization or prediction within the problem domain), but despite its good or perfect results it is very hard or impossible to infer how this ''knowledge'' was obtained, or how it emerged [3]. Put another way – correlations are discovered, while causalities may remain hidden. And transparency in this case would be very helpful as it would enable insights into the stages of conceptualization, and knowledge formation, within ML operations. This would make the gathered knowledge transferable to humans and other ML systems. In the case of DNN some efforts have produced promising insights into how concepts emerge in DNNs and evolve along their layers. However, these are still quite early stage results [3], [13].

The above insights into ML functioning will be referred to as functional transparency. Within the scope of this paper we will define another kind of transparency, and this will be referred to as data transparency. Although this latter kind of transparency is coupled with functional transparency, it still differs from it. Assuming the same ML architecture (algorithm), different sets of data will lead to different results. Moreover, even the same set of data represented differently, or pre-processed in another way (like tagging and cleaning) may lead to different results. Finally, different ML architectures may use the same data set, but may require different pre-processing activities. In brief, data transparency for security is about data sets standardization.

Clearly, data transparency is an important problem area if general ML explainability is to be achieved.

III. ML-BASED IoT SECURITY – FROM NOVEL ARCHITECTURE AND ALGORITHMS TO ENHANCED TRANSPARENCY

This section first presents the new, layered ML architecture for security provisioning in IoT networks, followed by subsections presenting a new algorithm and solutions for functional and data transparency. The presented solutions extend recent, rather rare research efforts in this specific area. One of the most similar types of research to ours can be found in [14], which focuses on cloud and edge-based solutions for anomaly detection. However, the solutions in this paper make a further push towards IoT hubs, making them independent of clouds, and therefore very suitable for more autonomous settings.

A. ARCHITECTURAL APPROACH

Based on the IoT network architecture and its integration with the internet (see Fig. 1 and Fig. 2) the following can be concluded. The IoT part, where the hubs perform central security activities, is computationally weak with constrained energy resources. These resources are now further deployed to run light ML solutions. As opposed to hubs, edge routers have considerably more resources and can be therefore used to run more precise, but also more resource demanding ML solutions. This leads to ML level one, and ML level two deployments, as presented in Fig. 3. This architecture reflects the networking reference models principle, where an implementation at one level (layer) is independent of the implementation at another level, while together they provide the required functionality through, e.g. service network access points. In our case this means the possibility of deploying (statistically) independent anomalies detection approaches, while their outputs can be combined via APIs with standardized data exchange formats to improve the results by following the basic laws of statistics.

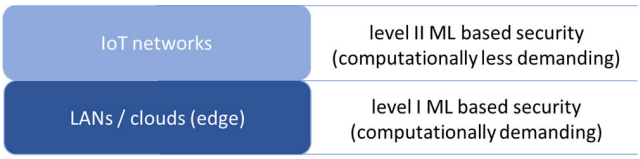


FIGURE 3. Two levels ML based anomaly detection for IoT networks.

So, for level I ML implementations, algorithms like DNN can be deployed (this part belongs to the ''full-blown part'' of traditional computing environments). These solutions, located at edge gateways, also have a better ability to store and use larger data sets needed for their training. Clearly, it is level I (i.e. edge) gateways in the given architecture that operate on gathered data flows from all hubs, which they are also able to store.

Hubs deploy level II ML based solutions for anomaly detection. Due to their nature hubs are able to deploy simpler algorithms such as those with unsupervised learning for clustering, or with supervised learning with linear regression. Unsupervised learning is more appropriate, because hubs are not able to store (large) flows and have to learn on the fly as much as possible. In short, the pipeline nature of processing and a reduced amount of data favors unsupervised learning on streams.

Therefore, both levels' solutions are expected to operate on flows, which (as already mentioned) is most common approach used today. Furthermore, this two stages architecture has another advantage. Although level II implementations are expected to have (notably) weaker performance than level I implementations (in terms of precision and recall values), this architecture actually presents a cascade of two

statistically independent ML implementations (e.g. unsupervised clustering at level II and DNN at level I). Consequently, the results from both points can be combined in a straightforward way taking into account the principles of statistics. Precision and recall (which corresponds to the true positive rate for classification), specificity (i.e. true negative rate) and F1-score are defined as follows are defined as follows [15]:

$$\begin{aligned} precision &= truePositives/(truePositives + falsePositives) \\ recall &= truePositives/(truePositives + falseNegatives) \\ specificity &= trueNegatives/(trueNegatives + falsePositives) \\ F1\text{-}score &= truePositives/(truePositives \\ &\quad + 0.5(falsePostives + falseNegatives)) \end{aligned} \tag{1}$$

When it comes to anomaly detection, recall has priority, because we in principle want as high a proportion of correctly identified positives as possible. Clearly, from the operational point of view good precision is also desired, because poor precision exhausts network operators with false alarms if it is low (but due to the nature of ML systems, these two requirements of recall and precision are often in tension). So, by taking into account the basic laws of statistics, cascaded, two-level ML applications can improve recall by reducing false negatives. In such settings even weaker ML implementations can notably improve the overall performance of anomaly detection solutions.

Therefore, when anomaly detection implementations consist of serially linked computationally weak and computationally strong ML implementations, security can be considerably improved, if each implementation deploys statistically independent procedures on the analyzed set of data. For the sake of simplicity suppose that each ML application has a false negatives rate of 40%. Running both strong and weak ML applications on the same set of data, the probability of false negatives is reduced to 16%. Therefore, even if the full-blown level I implementation has a false negative rate FN_1, and the weak level II implementation only FN_2, the overall false rate is improved as $FN_1{}^*FN_2$. Indeed, improving false negative rates with cascaded classifiers found its way into practical implementations with e-mail spam filters more than a decade ago [16].

B. NEW LIGHTWEIGHT ML LEVEL II IMPLEMENTATION

A new lightweight ML level II implementation called Profile and Hierarchical Incremental Clustering-based Anomaly Detection (PHICAD) is described next. Its pseudocode is given in Fig. 4, while its detailed architecture and one example detection run are presented in Fig. 5 and Fig. 6 in the appendix.

Compared to more general supervised and batch-based approaches, PHICAD has the following operational advantages. A potentially infinite data stream of raw network flows is represented by the input S_{flow}, where each new flow F_i is incrementally and efficiently preprocessed, analyzed and then discarded. It is unsupervised, so there is no need for a prerequisite knowledgebase of anomalous flows. It works on

```
Function PHICAD(S_flow):
 for F in S_flow do
  P_ipsrc, P_ipdst ← findSrcAndDstProfiles(F_i);
  for P in [P_ipsrc, P_ipdst] do
   T_i ← transform(F_i);
   V_i ← normalize(T_i);
   I_new ← createCF(T_i);
   IN_i ← getRootCFNode(P_tree);
   repeat
    IN_i+1 ← getClosestChildCFNode(I_new, IN_i);
   until IN_i+1 is not null;
   I_closest_leaf ← getClosestLeafCF(I_new, IN_i);
   A_i[] ← prepareEmptyDetectionReport();
   if d(I_new, I_closest_leaf) > T_ε
    r_0 ← checkDistance(I_i,centroid(I_neighbor[]));
    A_i[0] ← discardTooFrequentDetections(r_0);
    addToLeafCF(I_new, I_closest_leaf);
   else
    r_1 ← checkDistance(V_i, centroid(I_i));
    r_2 ← checkCentroidChanges(centroid(I_i));
    A_i[1] ← discardTooFrequentDetections(r_1);
    A_i[2] ← discardTooFrequentDetections(r_2);
    A_i[3] ← checkClusterSize(I_i, I_all[]);
    addToLeafCFNode(I_new, IN_i);
   end
   reportPredictions(A_i[]);
  end
 end
end
```

FIGURE 4. PHICAD algorithm high-level pseudocode.

the assumption that a large number of frequent flows represent the ''normal'' activity of the network and small number of infrequent and abnormal flows present ''anomalous'' activity. This also enables detection of new and unknown anomalies. It uses only eleven basic network flows features that can be provided by almost every flow collector available. Finally, most of the computation is done inside profiles, which are independent and can be run in parallel.

PHICAD aggregates flows into profiles, where each profile represents a model of network activity of a single network entity determined by its IP address. Each profile holds two network activity models, one for the flows where the profiles address is the source address, and the other where the profile address is the destination address. Each flow is sent to two profiles: the first determined by the flow's source IP address P_{ipsrc} and the second determined by the flow's destination address P_{ipdst}.

Inside a profile, the flows are first transformed into vectors T_i where all flow parameters (except timestamp and destination IP address) are directly represented by a single value, and the latter two are expanded to multiple values. Destination IPv4 address 8-bit parts are expanded into 4 integer values and a timestamp: days, hours, minutes, seconds, which are each represented by a sine and cosine value (see Fig. 5). This enables us to model the time difference between two timestamps, where the values in between jump through the minimum/maximum value barrier, which is a consequence of the periodicity. This means that the time difference between two timestamps, where the values in between 22:59 and

23:00 is one minute even though the values that present minutes changed from 59 to 0. Consequently, a flow with 11 features becomes a vector with 21 values.

After transformation, all values in the vector are normalized (V_i), where the minimum and maximum values are easily determined for all the vector values except for the maximum values of forward and backward packets and bytes. These four values were set at a threshold that covers 99.9% of all flows, while the 0.1% of flows with really large values are truncated to this threshold, which still enables us to discriminate between flows with values close to the minimum.

The models are built incrementally using a hierarchical clustering algorithm based on the BIRCH algorithm [17] (the process takes place on the hubs). The clustering algorithm continuously builds a height balanced tree, where each tree node holds cluster features that aggregate the nodes below it and the leaf nodes hold cluster features that present clusters. Each cluster feature stores only the number and sum of the fading weights of aggregated vectors, linear sum, and square sum for each dimension of the clustered vectors in this node, and not actual full vectors. These values enable us to easily combine multiple nodes into one when the tree structure is rearranged to improve the model and adapt to changes by reducing the importance of old vectors over time with the use of a fading function [18].

New vector V_i is first packed into a new cluster feature I_{new} and starts clustering at the root node IN_i of the tree. It descends through the closest child node IN_{i+1} on each level until it reaches the leaf node without children, updating the cluster features on its descent. If the closest leaf clustering feature $I_{closest\_leaf}$ is still too far ($d(I_{new}, I_{closest\_leaf}) > T_\varepsilon$), then I_{new} is added to the leaf node. For calculating the distance between cluster features and nodes Euclidean distance is used.

When I_{new} is clustered into a closest leaf cluster feature $I_{closest\_leaf}$, the following three different approaches are used to test if an anomaly has occurred in each dimension separately:

1. We track the $I_{closest\_leaf}$ centroid changes over time with adaptive window approach called ADWIN [19] for each of the dimensions. A potential anomaly in a window is detected when the harmonic means of the two sub-windows differ for more than a dynamic threshold derived from Hoeffding's inequality.
2. If the distance from $I_{closest\_leaf}$ centroid to I_{new} is greater than three standard deviations calculated from all the previously aggregated vectors in $I_{closest\_leaf}$, then we have detected a potential anomaly.
3. Every time $I_{closest\_leaf}$ is updated, its size is compared to all other leaf nodes. If its size is smaller than the half of the harmonic mean of all the leaf nodes sizes in a tree, then this cluster can present a potential anomaly.

Otherwise, I_{new} is added to the leaf node IN_{i+1}, where we check the distance of a I_{new} to the centroid of all the neighboring leaf nodes. If the distance is greater than three standard deviations, calculated from all the previously aggregated vectors in neighboring leaf nodes, then we have detected a potential anomaly.

Each test has a short-term memory for each of the dimensions that stores the times when changes were detected in the last 1000 updates to the profile. Fewer than 5% of the updates to the profile will be flagged as final anomaly detections. This is aligned with our assumption that anomalies are rear, abnormal events. The pseudocode for the above description is given in Fig. 4.

For a better understanding, a sample detection is shown in Fig. 6 in the appendix. It starts with the raw flow data, which is sent to the appropriate profile on the basis of the flow source IP address. The flow is then transformed and normalized and packaged into a cluster feature. The cluster feature descends through the tree until it comes to the leaf node, selecting the closest child node at each step. In the leaf node the new cluster feature is too far from other seven cluster features to be clustered into them.

We initiate the anomaly check by calculating the distance centroid of all other seven cluster features and the centroid of the new cluster feature, and if the distance is larger than three times the standard deviation of all the vectors aggregated in the seven other cluster features, then we have detected a change (this happens for the parameters flow duration, forward and backward packets, and bytes).

Next, a check is done if detections at those parameters were less frequent than 0.5% in the last 1,000 updates to the cluster. All of them passed the short-term memory check, so this flow is now flagged as anomalous. The detection is based on a large increase in forward and backward packets and bytes values which corresponds to the DoS attack that this flow presents. Consequently, PHICAD enables us to see why a certain flow was flagged as anomalous and we can transform the values in cluster features back to raw values for an even easier representation and understanding at any time.

C. PHICAD EVALUAITON

PHICAD has been evaluated using ISCX-IDS-2012 [20] and CIC-IDS-2017 [21] datasets, which consist of network flows for normal traffic and some of the most common network threats (DoS, DDoS, port scan, brute force, Heartbleed, web attacks, infiltration, bot net). These datasets were chronologically ordered, all non-TCP or UDP flows were discarded and then run through PHICAD algorithm in a single pass like a data stream. If a detection triggers inside a group of flows that present an anomaly, all those flows are classified as anomalous.

The anomaly detection performance and execution times of PHICAD are presented in Table 1 and Table 2. Overall, it achieves good performance with a F1-score of 0.81 for the ISCX-IDS-2012 dataset and 0.91 for the CIC-IDS2017 dataset. The execution times on a PC are also good, where we analyze roughly 20,000 flows per second, which makes PHICAD suitable for smaller IoT networks, where network traffic and computational resources are limited.

TABLE 1. The performance and execution times of PHICAD.

	ISCX-IDS-2012	CIC-IDS-2017
No. of flows	2,063,302	2,830,743
No. of anomalies	68,615	557,646
No. of detected anomalies	59,428	535,684
Precision	0.874	0.987
Recall	0.759	0.845
Specificity	0.999	0.997
F1-score	0.813	0.911
Execution [s]	99	142

TABLE 2. The performance and execution times of PHICAD for each type of anomaly in the CIC-IDS-2017 dataset.

	Precision	Recall	F1	Execution [s]
Brute force	0.29	0.12	0.17	29
DoS	0.99	0.98	0.99	108
Web attacks	0.68	0.98	0.81	10
Infiltration	0.02	0.04	0.03	17
Bot net	0.11	0.06	0.05	11
Port scan	0.97	0.99	0.98	26
DDoS	0.91	0.65	0.76	14

TABLE 3. Anomalies in CIC-IDS-2017 dataset.

	No. of flows		
	All	Anomalous	Detected anomalous
Brute force	445,909	13,835	3,002
DoS	692,703	252,672	250,065
Web attacks	170,366	2,180	2,146
Infiltration	288,602	36	9
Bot net	191,033	1,966	87
Port scan	286,467	158,930	158,557
DDoS	225,745	128,027	83,709

We focused on the CIC-IDS-2017 dataset because it is newer and because it represents multiple different types of network attacks, and not only DDoS. This dataset comprises eight files, where one contains only normal traffic and each of the other seven files contains normal traffic with a specific type of anomaly mixed in. Table 3 shows the number of all flows, anomalous flows and detected flows for each type of anomaly.

Next, Table 3 also shows the performance that was achieved with only a single file as the input stream. These figures prove that it achieves good performance for detecting denial of service, distributed denial of service, port scan and web attacks, where anomalous patterns clearly differ from normal activity patterns.

However, it should be added that the algorithm has issues with detecting anomalies, which do not exhibit a clear change in parameter values, like for example an Infiltration attack on the application layer.

Table 4 shows the PHICAD's performance compared to commonly used supervised approaches with a CIC-IDS-2017 dataset. It achieves the best precision, good recall, as well as a competitive F1-score and execution time. However, we have to take into account the operating advantages of PHICAD such as unsupervised learning, incremental execution, low use of computing resources, the use of only raw flow parameters that every network appliance can provide, no need for feature pre-selection before running the analysis and no parameters to set.

TABLE 4. The performance and execution times of PHICAD compared to other approaches for the CIC-IDS-2017 dataset.

	Precision	Recall	F1	Execution [s]
K-nearest neighbors	0.96	0.96	0.96	1908
Random forest	0.98	0.97	0.97	74
ID3	0.98	0.98	0.98	235
Adaboost	0.77	0.84	0.77	1126
Multilayer perceptron	0.77	0.83	0.76	576
Naïve-Bayes	0.88	0.04	0.04	15
Quadratic discr. analysis	0.97	0.88	0.92	19
PHICAD	0.99	0.84	0.91	142

Clearly, the proposed solution contributes to both the functional and data transparency. As to functional transparency, using light level II approaches inherently leads to better explainability. For example, linear regression is a well understood principle. Therefore, using it in ML helps to enable at least partial explainability of what ML systems have learnt.

Something similar holds true for unsupervised clustering. In PHICAD algorithm we analyze the raw flow parameters that are just transformed and normalized for the ease of use and can be rolled back to their raw values at any time. The clusters and distances in our model are just aggregates of the raw values and present a group of similar flows or distance between them. Furthermore, anomaly detection is based on a comparison of mean values, standard deviations, Euclidean distances or simple ratios. Therefore, the decision as to why a certain flow has been flagged as an anomaly is explainable, due to using simple and therefore explainable functions in a space defined by the flow parameters.

As to data transparency, the following, which we refer to as raw data sets alignment, needs to become a standard practice. Many sources of data are aiming at the same kind of deployment, and / or the same setting within this deployment. Therefore, the data elements and their types need to be defined in a standardized way, such as by JSON [22], or XML [23]. XML seems a better alternative due to complete data managements enabled via XML schemas, standardized data transformations support, and semantics support, while JSON focuses primarily on programming languages. When a standardized XML schema is available, the most extensive data set structure is defined. Then for any subsequent oper-

ations like cleaning and tagging, XML transformations can be used and transform templates stored in a data repository together with the very raw data that they are related to. This way all the training and testing data from their very raw form to the final real-field application would remain transparent.

Therefore, for level II data streams acquisition and processing in IoT settings the following schema is proposed and used in our implementation:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="flow">
 <xs:complexType>
 <xs:sequence>
  <xs:element name="sourceIP" type="xs:string"/>
  <xs:element name="sourcePort" type="xs:decimal"/>
 <xs:element name="destinationIP" type="xs:string"/>
  <xs:element name="destinationPort"
 type="xs:decimal"/>
  <xs:element name="protocol" type="xs:decimal"/>
 <xs:element name="timestamp" type="xs:datetime"/>
  <xs:element name="duration" type="xs:decimal"/>
  <xs:element name="forwardPackets"
 type="xs:decimal"/>
  <xs:element name="backwardPackets"
 type="xs:decimal"/>
  <xs:element name="forwardBytes"
 type="xs:decimal"/>
  <xs:element name="backwardBytes"
 type="xs:decimal"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

IV. DISCUSSION

First, although the presented solutions have wider deployments potential, they are particularly well suited for IoT networks. These networks are small and weak, but numerous. Unfortunately, this is a good basis for distributed denial of service attacks with one of the most disastrous kind of attacks being flooding attacks. Mirai was exactly based on exploiting IoT massiveness [24]. Furthermore, anomalies at the level of IoT networks (like body area networks) can be the result of ordinary malfunctioning, failures as such, which is also a safety issue. Consequently, the produced outliers are rather easy to be detected early, at the level II implementations, which is essential for body area networks in, e.g. medical settings [25].[1]

Second, despite the fact that streams-based anomaly detection is becoming a frequently deployed option, one has to be aware of its potential drawbacks, which may be subtle [11]. One group of drawbacks comprises measurement artefacts, so that traffic metadata do not resemble the original traffic anymore. These are the consequence of deployed devices deviations like under-dimensioned network links, race conditions, etc. Consequently, flows may contain time stamp errors, or gaps due to drops, etc. Another group are network artifacts, and they are the result of the deployed measurement setting in general. These comprise equipment calibration problems, retransmissions problems (which can be discriminated only at the transport layer, as counting them at the network layer where flows are formed gives false results), etc.

Third, data transparency problems need addressing soon. It is a basic fact that the more data there is the better the results of ML applications are. This has led to open data initiatives and available data sets are growing daily, being hosted by various repositories. One of the most widely recognized repositories with a variety of data from numerous domains is Kaggle (see https://www.kaggle.com/datasets). If data transparency issues are not addressed early enough, the exploding mass of raw data lacking adherence to some minimal principles for supporting data transparency may well influence explainability efforts in general.

Last but not least, even ML implementations themselves may be subject to security attacks as described in [26]. For example, even apparently minor modifications of submitted data like the proportion of a white background compared to the core content (e.g. the shape of a human body) can mislead an ML application and result in a wrong decision. But the very field of anomalies detection may pose other threats, which have yet to be identified. We believe that an appropriate synthetic dataset, which is high on the agenda of our future work, would counter this problem.

V. CONCLUSION

The recent penetration of IoT devices on the internet is fueling its growth and contributing to a data explosion. IoT networks, while weak in terms of available computing and energy resources, are a good target for attacks. The traditional security arsenal has recently been bolstered by ML applications, and IoT networks will be no exception.

However, these networks have many specific requirements, mostly due to limited energy availability and stringent computing resources. This results in limitations for traditional ML-based security approaches, in particular for anomaly detection. In order to achieve good results, appropriate ML algorithms have to be designed and deployed in appropriate architectural settings, which is why we have presented a new suitable architectural approach and corresponding ML solution in this paper. Furthermore, to enable a better understanding of the events behind detected anomalies, better insights into ML outputs are needed. Such explainability of the results leads to their transparency and a better understanding of the observed anomalies. This requirement also determines the most suitable ML approaches like clustering algorithms, which is the case in this paper.

Summing up, this paper presents a new approach to anomalies detection in IoT networks as follows:

[1]This paper also provides the details about previous work related to this research with additional references.

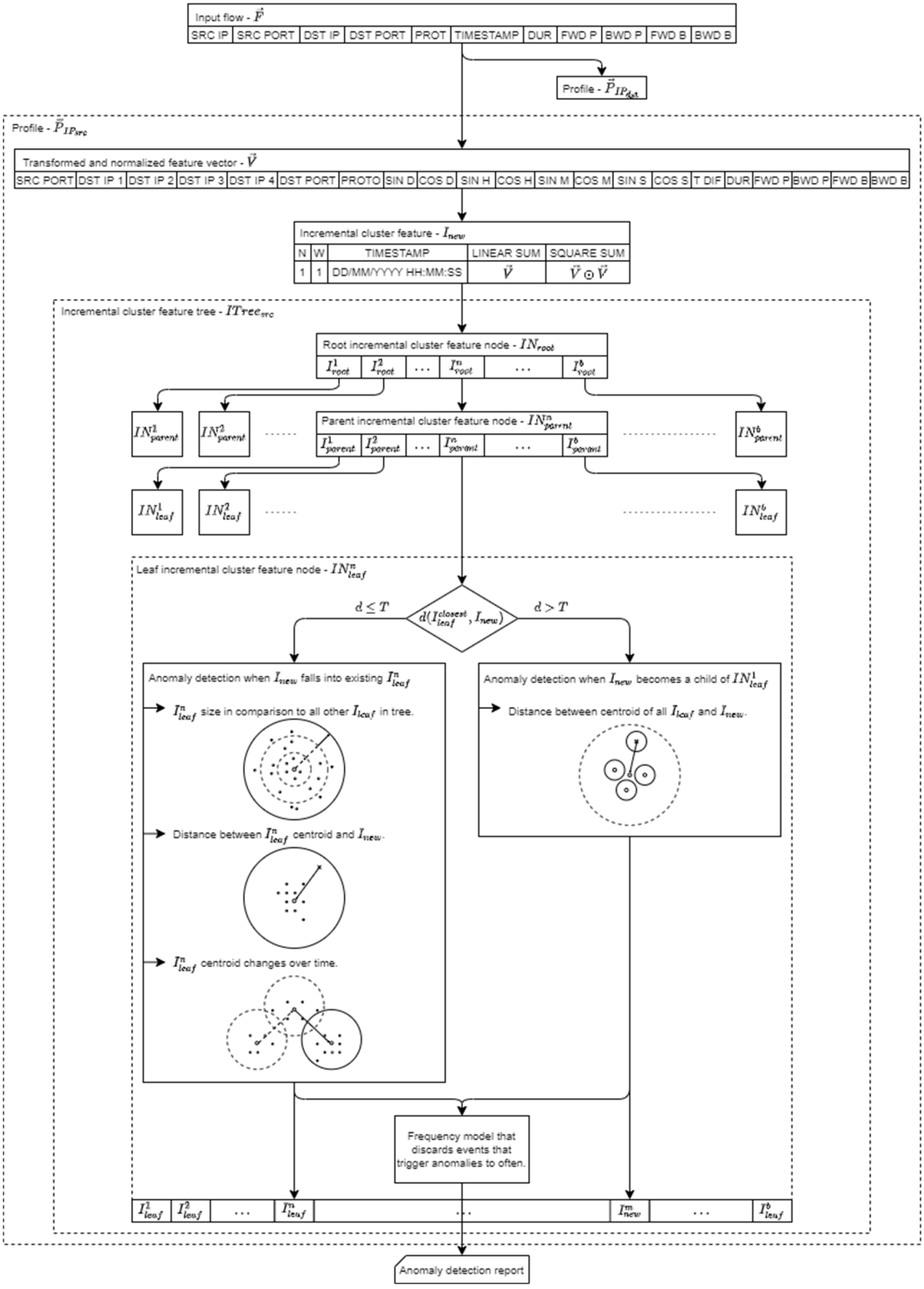


FIGURE 5. PHICAD algorithm for anomalies detection in IoT settings.

- First, the general architecture reflects the networking reference models principles, where the implementation at one level (layer) is independent from the implementation at another level, while together they provide the required (or improved) functionality. In the case of ML solutions, this enables deployment of statistically

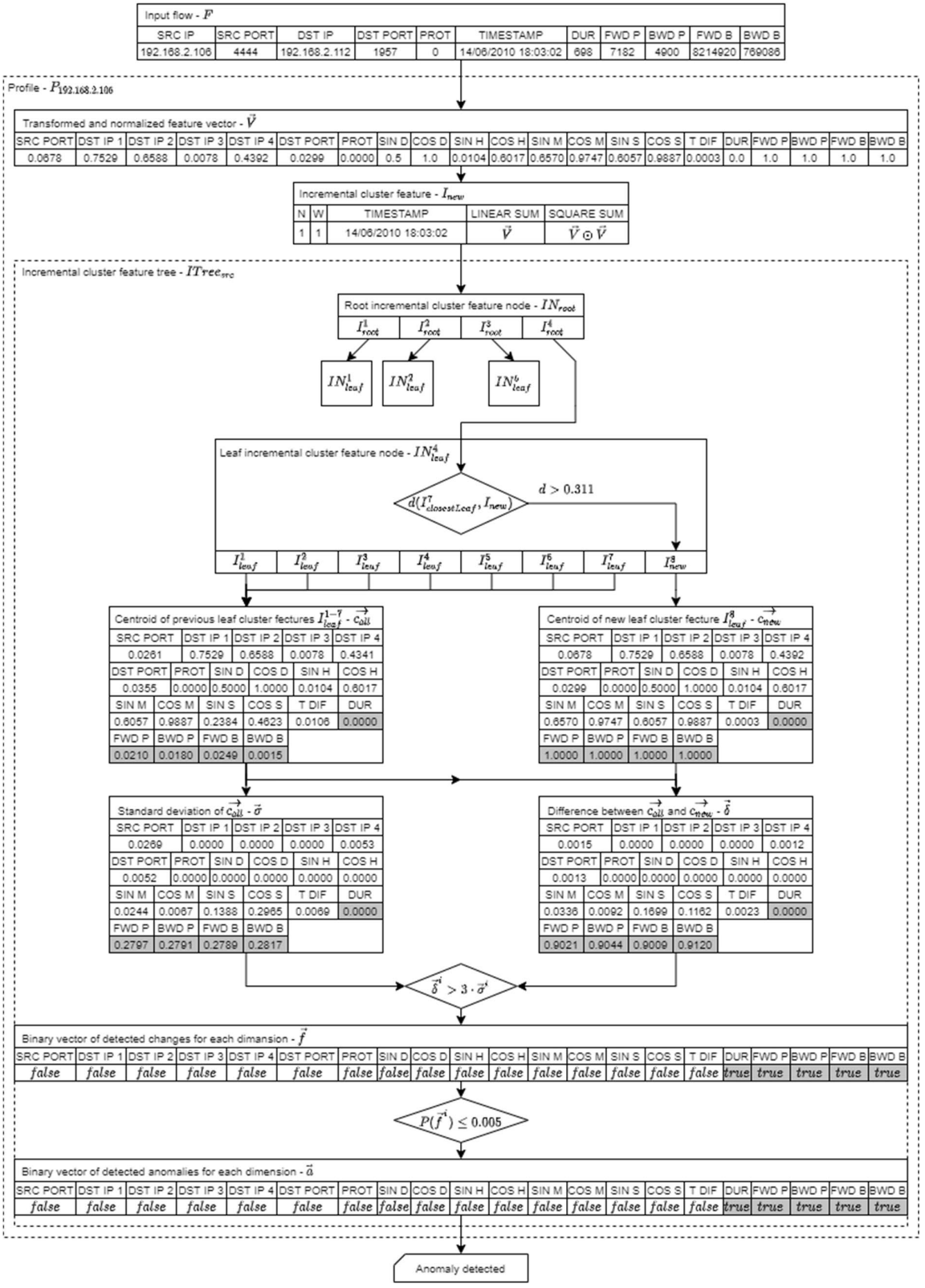


FIGURE 6. PHICAD algorithm run example.

independent anomalies detection engines, while their outputs can be combined due to standardized data exchange formats to improve the overall results by following the basic laws of statistics.

- Second, the developed ML algorithm is lightweight, and well suited for IoT devices. Despite being lightweight, its performance is aligned with the main ML based anomalies detection algorithms.

- Third, the developed solution is such that it also enables better functional and data transparency. Clearly, for a wider and better usability of ML based approaches, the results have to be explainable to the greatest possible extent.

Future work with PHICAD algorithm will address the addition of long-term windows to model specific timeframes that periodically repeat every day. PHICAD currently detects anomalies inside profiles, but the next step will be the analysis and comparison of multiple profiles to determine similar anomalies over multiple profiles, their evolution and the detection of new types of anomalies based on the changes in profiles groups.

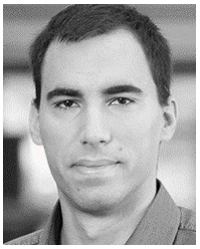
As to the presented approach in general, we expect that it will enable a step forward in security provisioning for IoT networks and stimulate further research in this area, which is of growing importance.

APPENDIX

See Figs. 5 and 6.

REFERENCES

[1] N. Gagliordi, (Nov. 27, 2018). IoT to drive growth in connected devices through 2022. Cisco. ZDNet. Accessed: May 11, 2020. [Online]. Available: https://www.zdnet.com/article/iot-to-drive-growth-in-connected-devices-through-2022-cisco/
[2] B. Naoufel, "Guidelines for modeling cyber-physical systems—A three-layered architecture for cyber physical systems," Siemens, Munich, Germany, Tech. Rep., 2017, doi: 10.13140/RG.2.2.21599.30881.
[3] W. Samek and A. Binder, "Tutorial on interpretable machine learning," in *Proc. MICCAI*, Granada, Spain, 2018. [Online]. Available: http://interpretable-ml.org/miccai2018tutorial/
[4] D. E. Comer, *Internetworking With TCP/IP*. Cham, Switzerland: Pearson, 2013.
[5] *Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*, ISO Standard 7498-1, Geneve, Switzerland, International Standards Organization, 1994.
[6] J. Hui and P. Thubert, *Compression Format for IPv6 Datagrams Over IEEE 802.15.4-Based Networks*, document RFC 6282, IETF, Reston, VA, USA, 2011.
[7] M. Felsche, A. Huhn, and H. Schwetlick, "Routing protocols for 6LoWPAN," in *IT Revolutions* (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), vol. 82, M. L. Reyes, J. M. F. Arias, J. J. G. de la Rosa, J. Langer, F. J. B. Outeiriño, and A. Moreno-Munñoz. Heidelberg, Germany: Springer, 2012.
[8] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, document RFC 7252, IETF, Reston, VA, USA, 2014.
[9] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and IPFIX," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2037–2064, 4th Quart., 2014.
[10] B. Claise, B. Trammell, and P. Aitken, *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*, document RFC 71011, IETF, Reston, VA, USA, 2013.
[11] R. Hofstede, A. Pras, A. Sperotto, and G. D. Rodosek, "Flow-based compromise detection: Lessons learned," *IEEE Security Privacy*, vol. 16, no. 1, pp. 82–89, Jan. 2018, doi: 10.1109/MSP.2018.1331021.
[12] K. Flanagan, E. Fallon, P. Connolly, and A. Awad, "NetFlow anomaly detection though parallel cluster density analysis in continuous time-series," in *Proc. Int. Conf. Wired/Wireless Internet Commun.*, 2017, pp. 221–232.
[13] W. Samek and R. K. Mueller, "Towards explainable artificial intelligence," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (Lecture Notes in Computer Science), vol. 11700, W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and R. K. Mueller, Eds. Cham, Switzerland: Springer, 2019, pp. 5–22. [Online]. Available: 10.1007/978-3-030-28954-6_1.
[14] F. Cauteruccio, G. Fortino, A. Guerrieri, A. Liotta, D. C. Mocanu, C. Perra, G. Terracina, and M. T. Vega, "Short-long term anomaly detection in wireless sensor networks based on machine learning and multi-parameterized edit distance," *Inf. Fusion*, vol. 52, pp. 13–30, Dec. 2019, doi: 10.1016/j.inffus.2018.11.010.
[15] K. M. Ting, "Precision and recall," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Cham, Switzerland: Springer, 2011, doi: 10.1007/978-0-387-30164-8_652.
[16] M. Takesue, "Cascaded simple filters for accurate and lightweight email-spam detection," in *Proc. 4th Int. Conf. Emerg. Secur. Inf., Syst. Technol.*, Jul. 2010, pp. 160–165.
[17] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM SIGMOD Rec.*, vol. 25, no. 2, pp. 103–114, Jun. 1996.
[18] C. C. Aaggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for projected clustering of high dimensional data streams," in *Proc. 13th Int. Conf. Very Large Data Bases*, 2004, pp. 852–863.
[19] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2007, pp. 443–448.
[20] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012.
[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018, pp. 108–116.
[22] *The JSON Data Interchange Syntax V2*, ECMA Standard 404, ECMA, Geneve, Switzerland, 2017.
[23] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, Eds., "Extensible markup language (XML) 1.0," W3C Consortium, W3C Recommendation, Cambridge, MA, USA, Tech. Rep. 2008REC-xml11-20060816, 2006.
[24] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.
[25] D. Trček, "Wireless sensors grouping proofs for medical care and ambient assisted-living deployment," *Sensors*, vol. 16, no. 1, p. 33, Jan. 2016, doi: 10.3390/s16010033.
[26] G. McGraw, R. Bonett, H. Figueroa, and V. Shepardson, "Security engineering for machine learning," *Computer*, vol. 52, no. 8, pp. 54–57, 2019.



ALEKS HUČ received the B.S. and M.S. degrees in computer science from the Faculty of Computer and Information Science, University of Ljubljana, Slovenia, in 2012 and 2015, respectively, where he is currently pursuing the Ph.D. degree in computer and information science. Since 2015, he has been involved in teaching of various undergraduate and graduate courses in computer and information sciences and collaborates on research as a member of the Laboratory of E-Media. His interests in computer and information science include machine learning (clustering and statistical methods) for the purpose of computer security. Since 2015, his research has been involved interdisciplinary fields related to the Internet of Things, computer security, machine learning, data mining, network technologies, sensors, web technologies, and food traceability processes.

DENIS TRČEK is currently a Full Professor with the Faculty of Computer and Information Science, University of Ljubljana, where he heads the Laboratory of E-Media. He has been involved in the fields of e-businesses, computer communications, and security and computational trust management for over twenty years. He has taken part in various EU and national projects mostly in government, banking, e-health, and cultural heritage sectors (projects under his supervision total approximately one million EUR). His bibliography includes over one hundred research works, including two monographs published by renowned publisher Springer Nature. He has served (or still serves) as a member for various international bodies and boards, such as the Management Board of the European Network and Information Security Agency. He has been, among other things, a Guest Researcher with Ludwig Maximillian Universität, Germany, and a Visiting Professor with Stanford University, USA.