# Identifying Security and Privacy Violation Rules in Trigger-Action IoT Platforms with NLP Models

Bernardo Breve, Gaetano Cimino, and Vincenzo Deufemia, *Member, IEEE*

***Abstract*—Trigger-Action platforms are systems that enable users to easily define, in terms of conditional rules, custom behaviors concerning Internet-of-Things (IoT) devices and web services. Unfortunately, although these tools stimulate the creativity of users in building automation, they may also introduce serious risks for the users. Indeed, trigger-action rules can lead to the possibility of users harming themselves, for example by unintentionally disclosing non-public information, or unwillingly exposing their smart environment to cyber-threats. In this paper, we propose to use Natural Language Processing (NLP) techniques to detect automation rules, defined within Trigger-Action IoT platforms, that potentially violate the security or privacy of the users. The proposed NLP-based models capture the semantic and contextual information of the trigger-action rules by applying classification techniques to different combinations of rule's features. We evaluate the proposed solution with the mainstream trigger-action platform, namely IFTTT, by training the NLP models with a dataset of 76,741 rules labeled by using an ensemble of three semi-supervised learning techniques. The experimental results demonstrate that the model based on BERT (Bidirectional Encoder Representations from Transformers) obtains the highest performances when trained on all features, achieving average Precision and Recall values between 88% and 93%. We also compare the achieved performances with those of a baseline system implementing information flow analysis.**

***Index Terms*—Privacy and security, Natural language processing, IoT platforms, Trigger-action rules.**

## I. Introduction

THE RISE of Internet-of-Things (IoT) technology initiated a new world of opportunities, especially in home environments. In fact, homes are becoming "smarter" as users increasingly purchase IoT devices such as connected cameras, smart locks, and smoke detectors, with the aim of building automation that would make their daily lives easier. IoT-based applications are built by programming a set of IoT devices to communicate with each other and perform certain tasks, e.g., voice-controlled cameras and remote-controlled door locks. To help users define interoperability behaviors between different smart devices and web services, several platforms have been defined and commercialized [1], [2]. Among them, the most popular are the *Trigger-Action IoT Platforms* (TAPs) [3] which empower users to define custom behaviors by means of conditional rules [4], [5].

Manuscript received 6 May 2022; revised 3 August 2022 and 6 September 2022; accepted November 13, 2022. This work has been supported by the Italian Ministry of University and Research (MUR) under grant PRIN 2017 "EMPATHY: Empowering People in deAling with internet of THings ecosYstems" (Progetti di Rilevante Interesse Nazionale – Bando 2017, Grant 2017MX9T7H). (*Corresponding author: Bernardo Breve.*)

The authors are with the Department of Computer Science, University of Salerno, I-84084 Fisciano (SA), Italy (e-mail: bbreve@unisa.it; gcimino@unisa.it; deufemia@unisa.it).

Samsung's SmartThings[1], Apple's HomeKit[2], IFTTT[3], and Zapier[4] are just a few examples of TAPs allowing users to create custom automation on devices and services in terms of Event-Condition-Action (ECA) rules [6]. The latter provide a suitable level of abstraction and are composed of a triggering *event*, the *conditions* that must be true when the event occurs, and the *action* that should be carried out [7]. The behaviors defined through ECA rules can address different aspects of a smart environment, for example, Fig. 1 shows an ECA rule whose behavior consists in the execution of the action "`open the shutters in the living room`" when the condition "`above 25° Celsius`" associated to the trigger "`the temperature of the house changes`" becomes true. A higher level of abstraction helps users to create their ECA rules more effectively and efficiently since the users do not need to deal with technical details that may be too complex to understand [8].

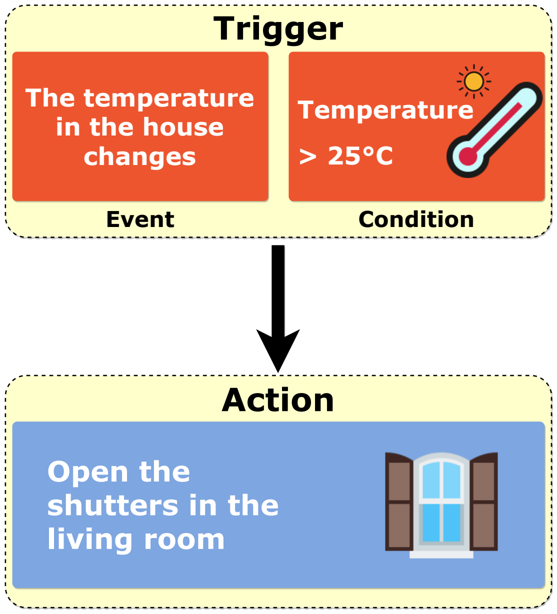
Trigger
The temperature in the house changes
Temperature > 25°C
Event
Condition
Action
Open the shutters in the living room

Fig. 1. An example of ECA rule.

However, when defining an ECA rule, users may create several serious risks for their privacy and/or the security of the smart environment [9], [10], [11], mainly due to their general

[1]https://www.smartthings.com
[2]https://www.apple.com/ios/home/
[3]https://www.ifttt.com
[4]https://zapier.com

inexperience and lack of technical knowledge. For example, a user who wishes to play music from his/her smartphone as soon as his/her Bluetooth earphones connect may define the rule: "`If a Bluetooth device is connected to the smartphone, then the smartphone plays the audio files contained in a folder`". This rule implies an important privacy disclosure as there may be a scenario where the selected folder contains personal audio files, and the smartphone connects to a Bluetooth speaker. Also, the rule in Fig. 1 could raise risks for the users. In fact, the opening of shutters is ruled only by the internal temperature of the smart house. In such a scenario, if the user forgets that the rule is active, shutters might open also on hot summer days when the house is empty, e.g. when the owners are on vacation, providing an entry point for thieves. Moreover, open windows represent a factor of risk for unsupervised children.

Several studies investigated the privacy and security risks of ECA rules highlighting that many rules are potentially dangerous [11], while the users have few concerns about these risks [12], and do not feel responsible probably because they assume that the companies producing the IoT device should guarantee their privacy [13]. However, very little research has been performed, so far, to address the problem of automatically identifying harmful ECA rules. In particular, Surbatovich *et al.* defined an information-flow lattice to analyze potential secrecy or integrity violations in ECA rules [11]. Paci *et al.* [14] proposed two approaches based on information flow analysis to detect rules that unintentionally violate users' privacy by sharing private photos. Other works focused on the automatic identification of undesired behaviors caused by rules' chain execution [15], [16], where a rule is automatically triggered by the action of another rule without user intervention.

In this paper, we investigate the efficiency of using Natural Language Processing (NLP) techniques to automatically identify potential security and privacy risks related to the execution of individual ECA rules. The approach exploits the capability of NLP models to semantically analyze the information of the rules, and infer how the type of trigger and action is related to the potential damage a rule might cause. In particular, the models analyze the triggers and actions associated with the rules, and the natural language textual descriptions provided by the creators. The latter represent an important resource for understanding the behavior of the rules, as demonstrated by the effectiveness of NLP-based analyzers that exploit these descriptions to generate executable code [17], and to infer the context in which the devices are involved [18].

We evaluate the considered NLP-based models on the IFTTT (If-This-Then-That) platform, which is the most popular for self-automation of IoT services, thanks to its ease of use, and the support for a large set of services and devices. We analyze a part of the dataset of IFTTT rules, which are named *applets*, crawled from the website and available on the Web [19]. We extract a set of 79,214 IFTTT applets, and classify them into four categories based on the type of damage they might cause. We select 2,473 rules for manual classification, and classify the remaining by using an ensemble of three semi-supervised learning techniques. Then, using the labeled dataset we train several classifiers on different combinations of the rule's features. The experimental results demonstrate that the model based on BERT, a pre-trained language model released by Google [20], achieves the highest Precision and Recall scores. Furthermore, we compare our approach with a baseline system implemented by using the information flow approach, which only focuses on the analysis of the event and the action chosen for building the rule [14], [11].

The main contributions of this paper are the following:

- Define a process for automatic labeling ECA rules with respect to security and privacy risks. The proposed strategy encompasses the application of an ensemble method to the predictions provided by different semi-supervised learning techniques.
- Make a dataset of ECA rules, crawled from the IFTTT platform and labeled with respect to security and privacy risks, publicly available in the additional material accompanying this paper (see Appendix A), so that others can advance work in the area.
- Provide an automatic, NLP-based approach for semantic-based and context-aware identification of security and privacy risks underlying ECA rules.
- Implement and evaluate the proposed approach on a mainstream trigger-action platform, i.e., IFTTT, showing that among the considered NLP-based models, the BERT-based one gives higher accuracy when trained on all rule's features.

A short, preliminary version of this article is available in [21], where we introduced a simple classification model trained on 2,000 manually labeled IFTTT applets. The rest of this paper is organized as follows. Section II reviews the literature concerning the privacy and security risks of trigger-action rules and their platforms. Section III summarizes the process of constructing NLP-based models for identifying rules violating security and privacy. Section IV provides details about the IFTTT platform and the structure of its rules. The construction of NLP-based models for IFTTT is detailed in Sections V and VI, while the experimental evaluation measuring their effectiveness is presented in Section VII. Finally, conclusions and further research are reported in Section VIII.

## II. Related Work

In this section, we discuss prior work for preserving the privacy and security of trigger-action IoT platforms and their users. We first discuss approaches that analyze and evaluate the potential risks of ECA rules, and then we review solutions that enhance platforms with mechanisms that prevent the disclosure of confidential information or malicious attacks.

### A. Analyzing Privacy and Security of ECA Rules

The study presented in [11] is the first that analyzes the privacy and security risks of IFTTT applets. In particular, Surbatovich *et al.* analyzed a dataset consisting of 19,323 IFTTT applets with a multi-level lattice that associates security labels to IFTTT triggers and actions. Information flow analysis is then performed to determine if a rule could involve a secrecy or integrity violation. The results highlighted that around

50% of the analyzed rules are potentially unsafe. They also manually categorized a subset of randomly selected applets according to the potential issue they can cause. About 60% of them were involved in a violation. The study presented in [10] refined the previous analysis by taking into consideration two factors while marking applets as harmful or not, i.e., the contexts in which the rules are applied, or the users' privacy preferences. The user study with 28 IFTTT users and 732 applets revealed that, with respect to the prior work, the number of harmful applets was significantly reduced by considering users' opinions.

In [14], Paci *et al.* focused on privacy issues related to sharing images via IFTTT applets. In particular, they introduced two prototypes, one for providing users with warnings about the privacy risks they may incur at design-time (when an applet is created), and one for providing warnings at run-time (when an applet is running). The first prototype considers the audience of the information, namely the "visibility", to report a privacy violation. In particular, the tool verifies whether sensitive information flows from a private trigger to a public or restricted action. The second prototype considers the "sensitivity" of the shared data. To this end, the tool exploits the Google "Vision" API to evaluate the sensitivity of a photo, reporting a privacy violation if a sensitive data item flows from a private trigger to a public or restricted action.

Other approaches focused on the problem of identifying unexpected behaviors potentially caused by ECA rules. McCall *et al.* [15] proposed SafeTAP, a tool that verifies through model checking whether the behavior of a new rule is affected by the existing ones [22]. Xiao *et al.* proposed A3ID, a tool for detecting implicit rule interferences, which occur when two or more rules are triggered simultaneously, causing contradictory effects on the environment [16]. A3ID uses NLP techniques to extract smart devices' knowledge (e.g., functionality, effect, and scope) from knowledge graphs. iRULER is a system proposed by Wang *et al.* for detecting different interferences conditions between trigger-action rules, such as action loop, where a rule is activated cyclically, or condition block, where the condition of a rule is unsatisfiable [3]. To this end, iRULER uses Satisfiability Modulo Theories solving [23] and model checking by operating on an abstracted information flow model inferred with NLP techniques. IoTMon [24] and SafeChain [25] are systems for identifying harmful attack chains produced by a combination of ECA rules. ProvThings is a tool that tracks data provenance for the purpose of providing explanations of rules' chain behaviors [26].

With respect to the approaches that identify harmful ECA rules through information flow analysis [11], [14], we propose the use of NLP techniques to extract semantic and contextual information from ECA rules, which are used to classify them according to the type of damage they could cause when activated. Moreover, while the approaches [15], [16], [3], [24], [25] aim to identify possible interferences/interactions between ECA rules, which could damage the smart environment, or affect the safety of the user, the proposed approach focuses on the identification of individual ECA rules that are potentially dangerous for the security and privacy.

## *B. Protecting TAPs from Privacy and Security Violations*

Xu *et al.* examined the possible leak of privacy information to which users may be exposed while using the main platforms for managing IoT devices and online services [27]. In particular, they analyzed how these platforms could reconstruct a user's behavior model through all the events for which s/he has defined a custom rule. To prevent this from happening, they introduced a process that filters and fuzzes the stored events. Bastys *et al.* demonstrated that IFTTT applets are vulnerable to attacks that could exfiltrate the private information of the users [28], and proposed two countermeasures. The former is based on an access control policy to prevent information flows from private sources to public sinks, the latter relies on a framework for monitoring the applet information flow over time, intending to identify what information the attackers might obtain from applet output and exploit them for future attacks.

Chiang *et al.* observed that TAP platforms are authorized to manipulate a lot of sensitive user information, and for this reason, they represent an attractive target for attackers [29]. To alleviate this problem, the authors proposed two platforms that can be integrated within a TAP to improve the privacy of users' data. The first aims to hide trigger information by sending fake information to the platforms, while the latter aims to preserve user privacy by masking the connection between the users and their data. Similarly, Chen *et al.* addressed the problem concerning the loss of users' sensitive data occurring when a TAP is compromised [30]. They proposed eTAP, an encrypted trigger-action platform capable of executing rules without accessing users' data in plaintext.

Fan *et al.* explored the possibility of attackers forcing rule executions with forged IoT devices or malicious events [31]. To face this issue, the authors proposed Ruledger, a ledger-based IoT platform that can be integrated within a TAP to guarantee the correct execution of rules.

IoTGUARD is a system that protects users from undesirable device states by monitoring trigger-action programs [32], blocking risky actions when integrity or confidentiality violations might happen. FlowFence is a framework that allows users to control their information flow once rules gained sensitive data access permissions [33]. SainT is a tool that identifies sensitive data flows by performing static analysis on information flow from sensitive sources to external sinks [34]. SOTERIA [35] and IotSan [36] are systems that apply model checking to control whether user security and safety properties are breached when using IoT platforms. IoT-Praetor is a system that exploits NLP techniques for extracting the interaction and communication behaviors of IoT devices from ECA rules, and comparing the obtained information to the actual behaviors detected at run-time [37].

The approach proposed in this paper is complementary with respect to those that safeguard the security and privacy of the users by protecting the information processed by TAPs [27], [29], [30], because it evaluates the risks by only analyzing the ECA rule's information. Similarly, the approaches proposed in [28], [31], [32], [33] use rule process monitoring techniques to identify potential damages at run-time, while our proposal identifies them at the time the ECA rules are defined. Moreover, while the approaches [34], [35], [36] are functionally dependent on the structure of an IoT program's source code, our proposal evaluates the behavior of an ECA rule without the need to analyze the source code. Finally, while [37] exploits NLP techniques to produce an intermediate representation of an ECA rule, we exploit NLP techniques for analyzing the semantic structure of an ECA rule with the aim of identifying potential damages.

## III. Construction of Classification Models for Identifying Harmful ECA Rules

This section describes the steps we perform to construct a classifier of harmful ECA rules. In particular, Fig. 2 depicts the process yielding the definition of effective supervised models for properly discriminating ECA rules according to possible classes of risk. More specifically, the process consists of three main phases:

(a) *Labeling ECA Rules*: this phase aims to prepare labeled datasets for classification models. Before carrying out this procedure, it is required to define the possible classes of risk for ECA rules, and the corresponding labels. Successively, each ECA rule of the input dataset has to be annotated with a suitable label. Since the manual labeling process is time-consuming, and the number of rules in the datasets might be very high, we propose to use a semi-automatic labeling strategy. In particular, we partition the original dataset of ECA rules into two subsets: a random small set of rules, which is labeled manually, and the set of the remaining rules, which is automatically labeled using semi-supervised classification models. The latter exploit the set of manually labeled rules to acquire sufficient knowledge for providing the labels to the rules contained in the larger set. Finally, an ensemble approach is employed to establish, among the labels provided by the semi-supervised models, which should be assigned to each rule, yielding a final large dataset of rules labeled according to the classes of risk.

(b) *Training ECA Rules Classification Models*: this phase aims to train classification models by using the dataset of labeled ECA rules, which is called *training set*. The features considered for training the models correspond to the components of the ECA rules revealing the context of use and the meaning of the rule, i.e., the trigger, the action, and the description of the rule behavior. These components are usually provided in textual form, so NLP techniques can be used to extract semantic information that can be exploited by classification models to identify and discriminate among the classes of risk. It is worth noting that the training set is most likely imbalanced, in fact, most of the rules do not provide damage to users, while some classes of damage are more frequent than others. To deal with the classification errors caused by imbalanced datasets, we consider the application of a weighted loss function [38], which weights the classification errors according to the number of rules available for each class. The best settings for the weighted loss function can be inferred by employing the stratified k-fold cross validation [39].

(c) *Testing ECA Rules Classification Models*: this phase aims to evaluate the performances of the classification models and, indirectly, the quality of the labels generated for the training set. This is performed by giving in input to the classification models a set of manually labeled ECA rules. The models' performances can be measured by using well-known metrics, such as *Precision*, *Recall*, *F1-score*, and *Accuracy*.

In the following sections, we describe how the considered process has been applied to a case study concerning the IFTTT platform.

## IV. The IFTTT Platform

IFTTT, which stands for "If This, Then That", is a trigger-action programming platform founded in 2010 by Linded Tibbets and Jesse Tane. Its visual interface allows users to define automation rules, which are called *IFTTT applets*, by specifying and configuring the two main components, i.e., the *trigger* and the *action*. The former defines the event(s) that activate the execution of the applet, whereas the latter corresponds to the operation carried out upon the applet triggering.

For creating an applet, the user has to first specify the service, also named *channel*, associated with the trigger component, e.g., a social network or the cloud service of an IoT device producer. After selecting the channel, the user has to select the trigger among a range of possible ones, completing the first section of the applet. The operation is then replicated for the action section, where the user can select the *channel* and its *action*. Depending on what trigger or action is selected, the user may be required to add further details to complete the definition of the rule's behavior, these details are called *fields* whose values can be, for example, the folder name where to upload a file on Google Drive, or which parameters tracked by a Fitbit device should be recorded. Finally, the user can specify a *title* and a *description* to remember the purpose of the applet and to make it easy for other users to understand. In fact, every created applet is automatically acquired by the IFTTT platform so that other users can immediately activate it without needing to create a new one.

The dataset used for training and testing the models for classifying harmful IFTTT applets was created by researchers from Indiana University Bloomington [19], through a crawling process on the `IFTTT.com` website from November 2016 to May 2017. During this period, the authors took every week a "snapshot" of the applets published on the platform at that moment. About 200 GB of data ($\sim$12 GB per snapshot) were collected during that period, corresponding to more than 300K unique applets. The researchers have kept the same structure of the IFTTT paradigm, that is the decomposition of the applet into trigger, action, and the corresponding channels. In particular, for each applet, the following information was retrieved: applet name, description, trigger, trigger channel, action, action channel, a counter indicating the number of users who have installed the applet, and other features.

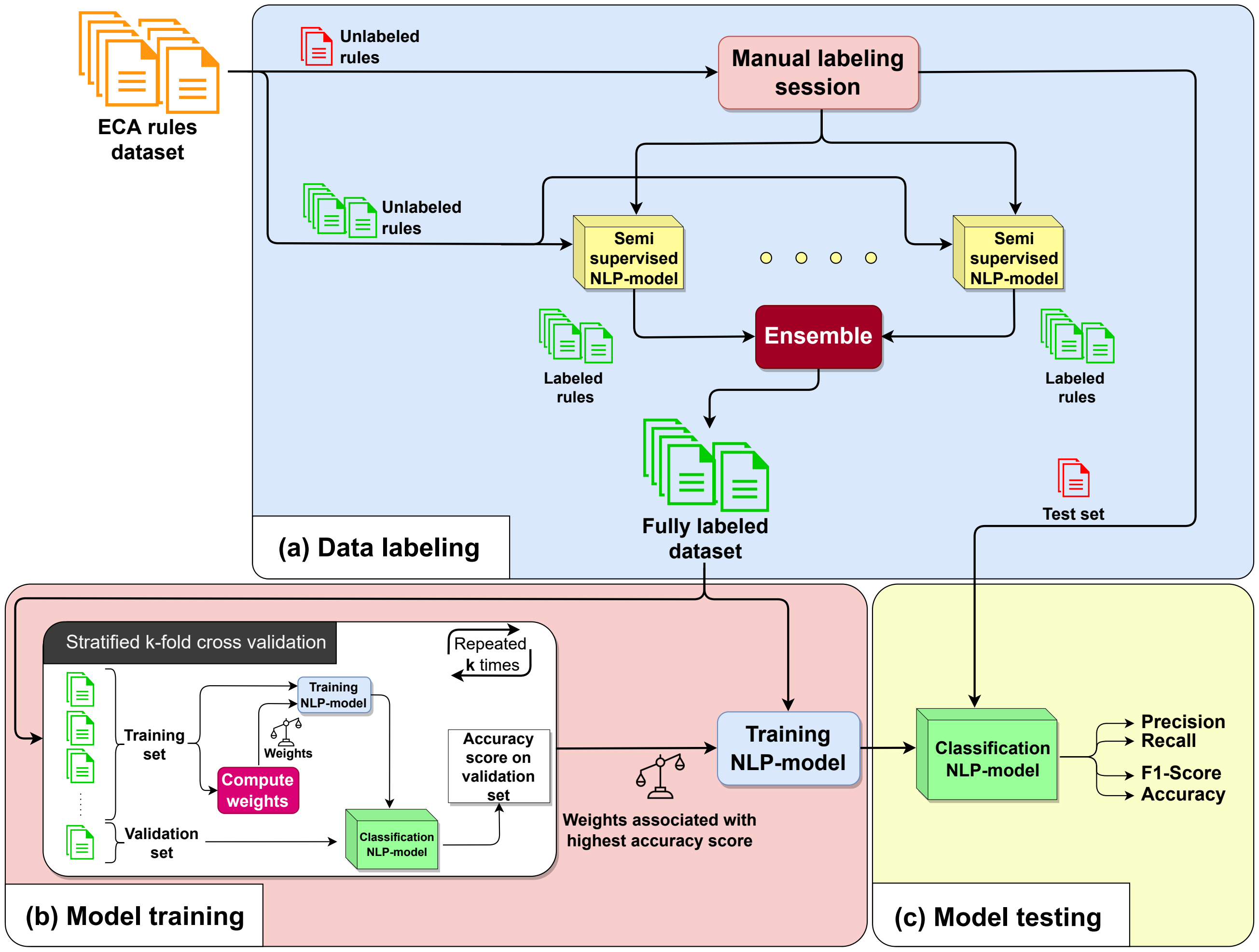

Fig. 2. The proposed process for constructing and evaluating NLP-based models for detecting harmful ECA rules.

In our work, we only consider the following applet features:

- Title: is a string containing the title of the applet;
- Desc: is a string containing a description of the applet's behavior;
- TriggerTitle: is a string representing the name of the trigger that “activates” the applet;
- TriggerChannelTitle: is a string representing the name of the trigger channel chosen by the user as a trigger for the applet;
- ActionTitle: is a string representing the name of the action to execute in response to the trigger;
- ActionChannelTitle: is a string representing the name of the action channel chosen as an action by the user.

It is worth noting that the first two features are continuous since they are defined by the applet's creator, whereas the remaining ones are discrete since they are automatically populated by IFTTT and can assume a finite number of values. As an example, the ECA rule shown in Fig. 1 could be specified by the following parameters:

- *Trigger channel*: `Netatmo Weather Station`
- *Trigger*: `Temperature rises above`
- *Action channel*: `Gewiss Smart Home IoT`
- *Action*: `Control your Shutter or Venetian`
- *Fields*: `Temperature threshold (25° in Fig. 1), Unit of measure (Celsius in Fig. 1)`
- *Title*: `Let some fresh air comes in your house when it gets too hot`
- *Description*: `If the temperature inside your house is above a certain threshold, automatically open the shutters.`

An accurate data cleaning process is performed on the initial dataset, which considerably reduces the number of applets. In particular, to obtain a uniform dataset in language, we use the LANGDETECT Python library to filter out the applets whose title and description are not written in English. Subsequently, we discard all applets without title or description or containing only numbers for these features. The resulting dataset consists of 116,825 applets, and is provided in the supplementary material (see Appendix A).

## V. Data Labeling

As said above, the rule labeling process is performed by applying a combination of manual labeling and automatic labeling with semi-supervised models, and an ensemble strategy. This allows us to perform extensive labeling of the dataset minimizing the manual effort. In this section, we go through all the phases yielding a fully labeled dataset with respect to the considered classes of risk.

### A. Categorizing Applets based on Security and Privacy Risks

We consider the work in [11] for categorizing the damages that could be inflicted by an applet on the user. An interesting result of the analysis performed in [11] is that not all applets need a third party to cause a risk. In fact, many applets are dangerous due to issues resulting from users' behaviors. For instance, at first sight, an applet having title "`Keep your Facebook and Twitter profile pictures in sync`" could not seem harmful. However, in a scenario where a user has a private Facebook profile, if s/he forgets that such an applet is active, unwanted photos will automatically be published to Twitter, where the audience might not be the same as the one on Facebook, causing possible embarrassment. In this case, an attacker does not need to intervene for providing damages, since is the user's behavior that generates harm to his/her own privacy. At the same time, other applets could violate users' privacy/security due to attackers exploiting the behavior of an applet to damage an IoT device or to disrupt online services.

In [11], by manually examining a set of applets, the potential damages they can cause are classified into four macro-categories:

1) *Innocuous:* causes no harm, i.e., an applet for which it is not possible to imagine a realistic harm. As an example, the applet: "`If I meet my daily step target, update a file with the statistics on my phone`" has no negative consequences because sensitive information is not shared with third parties, without causing embarrassing situations;
2) *Personal:* causes loss of sensitive data. This damage is self-inflicted since any damage is the result of the user behavior. As an example, the applet: "`If I take a new photo, then upload it on Instagram as a public photo`" could unintentionally leak sensitive information;
3) *Physical:* causes damage to physical health or goods. This damage is external as a third party can potentially inflict the damage. As an example, the applet: "`If the last family member leaves home, then turn off lights`" is dangerous since turning off the lights in a predictable way signals that the house is empty, making it easier for a thief to plan the right time for a theft;
4) *Cybersecurity:* causes interruption of an online service or distribution of malware. This damage is external too, as a third party can potentially inflict the damage. As an example, the applet: "`If there is a new email in your inbox with an attachment, then add the attached files to OneDrive`" could be used to spread malware to all devices synced with a OneDrive account. If a malicious attachment is propagated to all synced devices, it increases the likelihood that the file will be opened by the user.

The analysis of the results highlighted that the most common damage is personal, i.e., the one caused by mistakes of users and not from the involvement of a third party. Concerning damages potentially inflicted by an attacker, cybersecurity damages are found to be more frequent than physical ones.

According to the considered macro-categories of risk, we use the following classes for applet labeling: class *0* corresponds to "Innocuous" applets, class *1* to the applets underlying a "Personal" damage, class *2* to the applets which could lead to a "Physical" damage, and class *3* to the applets exposing "Cybersecurity" damages. In the following, we provide the details about the manual and the automatic labeling processes.

### B. Manual Applet Labeling

We apply the majority method for manually labeling the applets of the IFTTT dataset. In particular, the first and second authors are in charge of manually labeling the applets, while the third intervenes when there is no agreement. This phase leads to 492 class 0 applets, 296 class 1 applets, 105 class 2 applets, and 107 class 3 applets.

To balance the number of applets of each class, we define a process for selecting further applets to be labeled manually as shown in Fig. 3. For each labeled applet $a$, we build a spreadsheet containing all the unlabeled applets sorted in descending order based on their similarity with respect to $a$. The similarity is evaluated through a combination of vector semantics and similarity functions. The former is used to compute a vector representation of sentences, namely *sentence embeddings*, which takes into account their semantic meaning [40]. For computing the sentence embedding of each applet, we apply SentenceBERT [41] on the concatenation of the applet's title and description, whereas the cosine similarity function is used to compare the embeddings.

By manually reviewing each spreadsheet, we select and label a subset of applets having "similar" characteristics to those already labeled, but with at least a difference in the trigger, the action, and/or the involved channels. The resulting dataset of manually labeled applets consists of 503 class 3 applets, 502 class 2 applets, 501 class 1 applets, and 967 class 0 applets, for a total of 2,473 instances.

### C. Automatic Applet Labeling

To increase the number of labeled data, we devise a process combining different semi-supervised learning models with an ensemble strategy, as shown in Fig. 2. This section illustrates the three semi-supervised learning techniques we use to automatically label additional applets, namely Self Learning, Label Propagation, and Generative Adversarial Learning, and the ensemble strategy we adopt to generate the final labeled dataset.

*1) Self Learning:* It consists in turning any supervised classifier into a semi-supervised method by iteratively labeling the unlabeled data, and adding these predictions to the set of labeled data until the classifier converges [42]. More specifically, we implement the following Self Learning process:

(a) Train a classifier $C$ on the set of available labeled applets $A$;
(b) Use $C$ to make predictions on the set of unlabeled applets $U$;

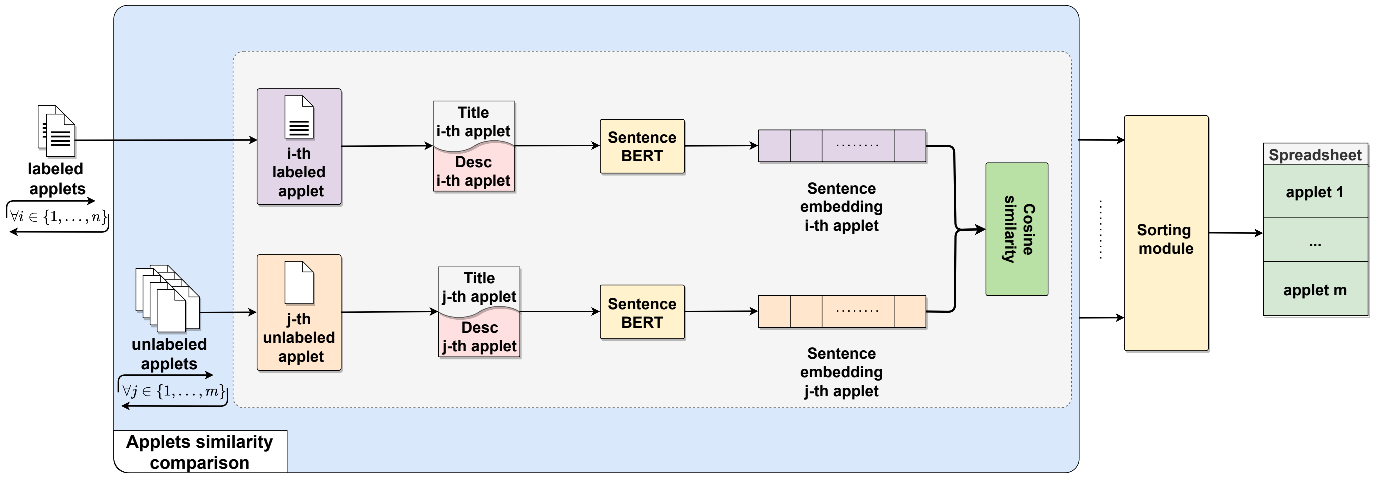

Fig. 3. Similarity comparison process of IFTTT applets based on SentenceBERT.

(c) Move from $U$ to $A$ the applets whose predictions satisfy a condition defined with a *confidence* parameter, which is specified a priori. These are called "*pseudo-labeled applets*" to distinguish them from the original labeled ones.

Steps (a)-(c) are repeated until one of the following convergence criteria is reached:

- The maximum number of iterations is reached;
- All predictions obtained on the applets of the set $U$ do not satisfy the condition in (c);
- The set $U$ is empty.

We set the maximum number of iterations to 5, the confidence parameter to 0.6, and use BERT as a classifier [20]. Moreover, we use the *softmax* function to compute the probabilities with which unlabeled applets are associated with each class. Thus, if the highest estimated probability is greater than 0.6 then the corresponding class is assigned to the applet, becoming a pseudo-labeled applet.

*2) Label Propagation:* It is an inference algorithm based on semi-supervised graphs [43]. The algorithm constructs a *similarity graph* that models the set of training data trying to propagate known labels across the edges of the graph, from labeled samples to samples for which the label is not available. Since the data are in textual form, we apply techniques for obtaining a vector representation of them. In particular, we use the sentence embeddings computed through SentenceBERT [41], which produces a 512-dimensional vector of the input sentences.

The graph construction process deals with converting the dataset $X$ into a graph $G$, where $X$ represents the input samples composed of the applets $x_1$, $x_2$, ..., $x_n$, and each applet $x_i$ in turn is represented by a 512-dimensional vector. Each applet is assigned to a node of the graph, and a weight $w_{i,j}$ is assigned to each edge connecting the pair of nodes *i* and *j*. To identify the similarity between two nodes, a matrix of weights $W$ is computed. To calculate the weights, it is necessary to employ a kernel. Among the possible applicable kernels, we use the *K-nearest neighbors*, which produces a fully connected graph represented in memory by a *sparse* matrix, and guarantees fast execution times. The obtained weights are used to compute the probabilities of propagating a label from a labeled node to an unlabeled one. In particular, the probabilities are used to assign soft labels to each node that can be interpreted as distributions over labels. Labeled nodes have probability 1 of belonging to one of the four classes (since their classes are known), while unlabeled nodes get their class from "neighboring" nodes. During the execution, these distributions are altered causing the change of the labels assigned to unlabeled applets. The process iterates until convergence, i.e., the probabilities do not change and the labels associated with unlabeled applets are no longer changed, or a fixed number of iterations is reached. We set the maximum number of iterations to 1,000.

The `LabelPropagation` class of the SCIKIT-LEARN Python library is used to implement the algorithm.

*3) Generative Adversarial Learning for Robust Text Classification with a Bunch of Labeled Examples:* It is an extension of the BERT model within the Generative Adversarial Network (GAN) framework allowing for the implementation of an effective semi-supervised learning schema [44]. This model allows training BERT on a limited number of labeled samples, with respect to a larger number of unlabeled samples. The fine-tuning phase of BERT is extended by introducing a Discriminator-Generator setting, where:

- The generator $G$ deals with the production of "fake" vector representations of sentences. In particular, $G$ produces "fake" samples by taking as input a 100-dimensional noise vector based on a Gaussian distribution;
- The discriminator $D$ is a BERT-based classifier that works on $k+1$ classes. In particular, $D$ receives as input either a fake vector generated by $G$, or a vector from real data embedded by BERT. The final layer of $D$ is a Softmax Output Layer producing a vector of size $k+1$, where $k$ is the number of classes in the training set.

$D$ has the role of classifying an instance as one of the $k$ classes related to the task of interest (in this case $k = 4$) and must recognize the instances generated by $G$ (to which it must associate the class $k+1$). In other words, it must classify whether the input is a real instance or not; if it estimates the input as a real instance, it must predict which class the input belongs to. On the other hand, $G$ must produce representations as similar as possible to the "real" instances. $G$ is penalized when $D$ correctly classifies an instance as fake.

We employed the PYTORCH interface of the TRANSFORMERS Python library of HUGGING FACE for implementing the model. We set the parameters of the model as follows:

- Batch size: 32;
- Number of hidden layers for the Generator: 1;
- Number of hidden layers for the Discriminator: 1;
- Size of the noise vectors: 100;
- Learning rate Discriminator: $5e-5$;
- Learning rate Generator: $5e-5$;
- Epsilon: $1e-8$.

Since $D$ must also be trained to discriminate real sentences from fake ones, we introduce an additional class to those of the task of interest; in particular, the identifier "4" has been associated with the "fake" class.

*4) Ensemble:* Starting from the three datasets of labeled applets obtained with the semi-supervised learning approaches, we apply an ensemble strategy to get a single dataset. In particular, we use a majority-vote method across the three different semi-supervised models, assuming that if two models agree, the prediction would be more accurate. Thus, only the applets for which at least two semi-supervised techniques agree on their class labels are included in the final dataset with that class. This allows us to obtain more consistent labels for the evaluated applets.

With this strategy we obtain a dataset containing 79,214 applets, where 56,236 belong to class 0, 16,344 to class 1, 3,433 to class 2, and 3,201 to class 3. Table I reports statistics about the service categories involved in the labeled dataset. For each service category, the table provides the percentage of involved services, and the percentage of applets whose triggers (actions, resp.) belong to a service within the category. We can observe that most of the services are for IoT devices, while triggers from social networking services are the most popular among the applets. On the other hand, almost half of the applets in the dataset have an action belonging to the "RSS feeds, online recommendation" category.

TABLE I
STATISTICS ABOUT THE SERVICES INVOLVED IN THE APPLETS OF THE LABELED DATASET

| Service category | % Services | % Triggers | % Actions |
|---|---|---|---|
| Smarthome devices (e.g., Light, thermostat) | 44.79% | 5.08% | 4.33% |
| Online service and content providers | 16.60% | 11.72% | 14.99% |
| Social networking, photo/video sharing | 8.21% | 35,78% | 18.32% |
| Smartphones (e.g., battery, NFC) | 5.56% | 11.09% | 2.62% |
| SMS, instant messaging, VoIP | 5.55% | 7.51% | 2.29% |
| RSS feeds, online recommendation | 4.11% | 2.24% | 44.34% |
| Smarthome hub (e.g., Samsung SmartThings) | 2.90% | 0.08% | 0.03% |
| Personal data & schedule manager | 2.90% | 7.11% | 0.64% |
| Wearables (e.g., smartwatch) | 2.66% | 0.37% | 0.84% |
| Cloud storage (e.g., Google Drive) | 2.17% | 9.35% | 1.01% |
| Time and location | 1.60% | 0.07% | 6.49% |
| Email | 1.45% | 9.45% | 4.01% |
| Other | 1.50% | 0.15% | 0.09% |

## VI. MODEL TRAINING

This section illustrates the models we implement for classifying the IFTTT applets, the techniques we adopt to solve the problem of imbalanced data in the final dataset, and the setup of the training phase. The models are implemented as Python modules, which are provided in the supplementary material (see Appendix A).

We consider two types of classifiers. The first is based on artificial neural networks (ANNs), and treats discrete features as numerical values, by using the label encoder technique [45], and continuous features as textual values, by using a word embedding technique [46], namely *Global Vectors for Word Representation* (GloVe) [47]. The latter is based on BERT, a pre-training model of Natural Language Processing, which uses deep bidirectional transformers to train a language representation model through a large number of data [20]. In this model, all features are treated as text.

The ANN-based and BERT-based models are trained considering three combinations of features:

| | |
|---|---|
| *combination 1*: | Title and Desc; |
| *combination 2*: | TriggerTitle, ActionTitle, ActionChannelTitle, and TriggerChannelTitle; |
| *combination 3*: | All features. |

Before training the models, a pre-processing phase is carried out to remove noise from the data. In particular, we perform the operation of tokenization, normalization, noise removal, and lemmatization on the textual values. Successively, the encoding phase converts the dataset into a format valid for the classifiers.

### A. Classification by Artificial Neural Networks

In the following, we illustrate the different architectures implemented for the ANN models. In particular, we consider a simple neural network (NN) when the features are discrete (combination 2), and the *Long Short-Term Memory* (LSTM) when continuous features are involved (combinations 1 and 3).

*1) First combination:* We train an LSTM model, named *LSTM-1c*, using the Title and Desc features. Their GloVe word embeddings are vector representations of textual data having a fixed length. Since the title and description of the applets have different lengths, an embedding with a longer length will be filled with zeros at the end (representing the so-called *padding*), if it is shorter, it will be truncated. To identify the most suitable embedding length, we analyze the distribution of the lengths of the dataset sentences obtained by concatenating Title and Desc. As shown in Fig. 4, most sentences are composed of 25 words or less, whereas the maximum length is 400. Using 25 as embedding length is unsuitable because many words may be lost. At the same time, setting the length to 400 leads to embedding with a lot of padding, which would not help the model to learn. Therefore, we set the length of the embeddings to 50, which corresponds to a good trade-off.

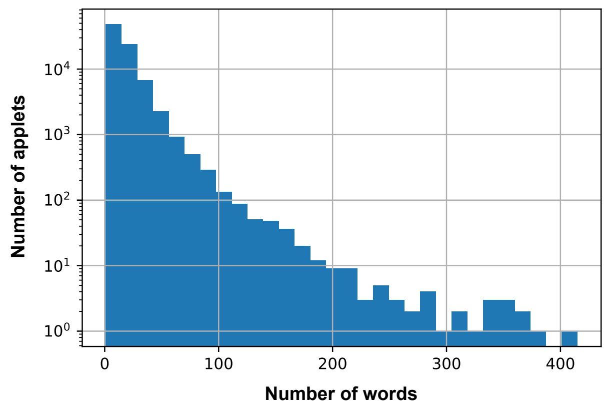

Fig. 4. Length distribution (in logarithmic scale) of the sentences obtained concatenating the Title and Desc features.

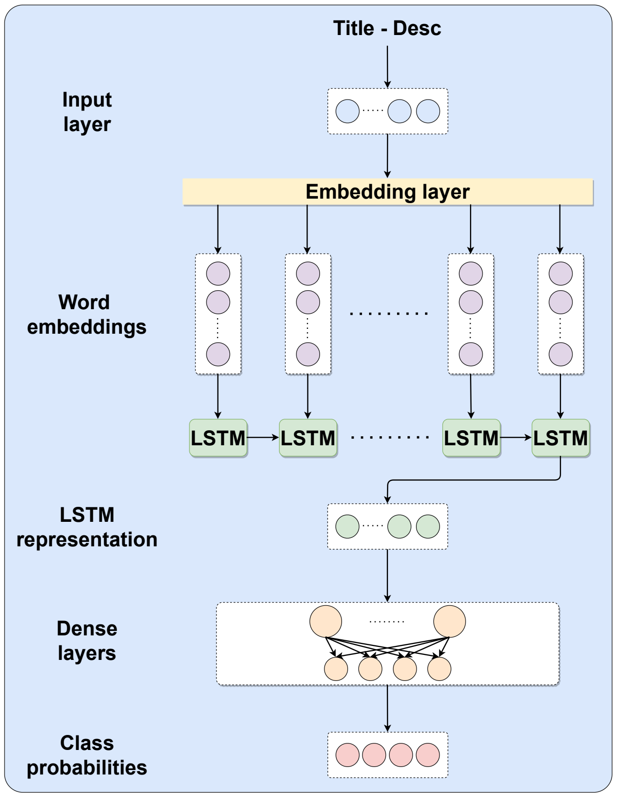

Fig. 5. The architecture of the LSTM-1c model.

Fig. 5 shows the architecture of the LSTM-1c model. Specifically, the *Input Layer* receives embeddings of 50 dimensions, then the *Embedding Layer* turns positive integers into dense vectors of fixed size. The latter requires the following parameters:

- The size of the dictionary (i.e., the number of distinct words in the input sentences);
- The size of the output vector, which is 50, as the dimension of the embeddings;
- The pre-trained embedding matrix specified using the *weights* attribute.

The *LSTM Layer* is composed of 70 neurons and uses the *tanh* (hyperbolic tangent) activation function, whose outputs are in the range [-1,1]. Subsequently, the *Dense Layer* is composed of 35 neurons and uses the *tanh* activation function. The values of the two parameters are obtained through a fine-tuning process.

The final Dense Layer acting as the *Output Layer* consists of 4 neurons and uses the *softmax* activation function. The number of neurons corresponds to the number of classes that the classifier has to predict. The LSTM-1c model is trained using 24 *epochs* and a *batch size* of 10.

*2) Second combination:* We train a simple Neural Network model, named *NN-2c*, using the discrete features of the dataset. For these features, it is possible to apply the *label encoder technique* to convert them into numerical ones. This technique simply assigns a unique numerical value to each categorical value that a feature can assume. In this way, we obtain well-structured data, which can be used as input for densely connected neural networks.

Fig. 6 shows the architecture of the NN-2c model. It is composed of an *Input Layer* that accepts an input with size 4 (i.e., the values of discrete features) and three *Dense Layers*. Two of them consist of 50 and 20 neurons, and use the *tanh* activation function. As for LSTM-1c, the final Dense Layer acts as the *Output Layer* and consists of 4 neurons, with the softmax as an activation function. The NN-2c model is trained using 30 *epochs* and a *batch size* of 16.

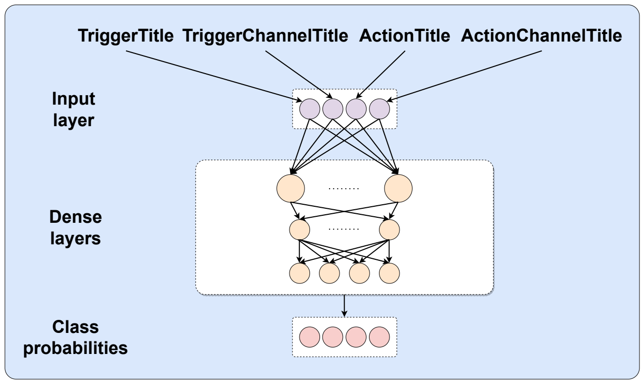

Fig. 6. The architecture of the NN-2c model.

*3) Third combination:* We train an LSTM model, named *LSTM-3c*, using all features of the dataset. The discrete features are converted from categorical to numeric and combined with the continuous ones. To efficiently handle the different types of inputs, i.e., textual (i.e. Title and Desc) and numerical (i.e. TriggerTitle, ActionTitle, ActionChannelTitle and TriggerChannelTitle), we define two sub-models: the former receives textual input encoded with GloVe, while the latter receives numerical input encoded with the label encoder technique.

Fig. 7 shows the network architecture. The first sub-model is composed of an *Input Layer*, which receives embeddings of 50 dimensions, an *Embedding Layer*, and an *LSTM Layer* characterized by 70 neurons and using the *tanh* activation function. Also, the second sub-model is composed of three layers. In particular, an *Input Layer* receiving the four numerical values of the discrete features, and two *Dense Layers*, composed of 50 and 20 neurons respectively, on which the *tanh* activation function is used. The output of the LSTM Layer of the first sub-model is concatenated to the output of the second Dense Layer of the second sub-model and used as input for another Dense Layer characterized by 10 neurons. Finally, the last Dense Layer acts as the *Output Layer* and is characterized by 4 neurons, corresponding to the classification classes. The LSTM-3c model is trained using 35 *epochs* and a *batch size* of 10.

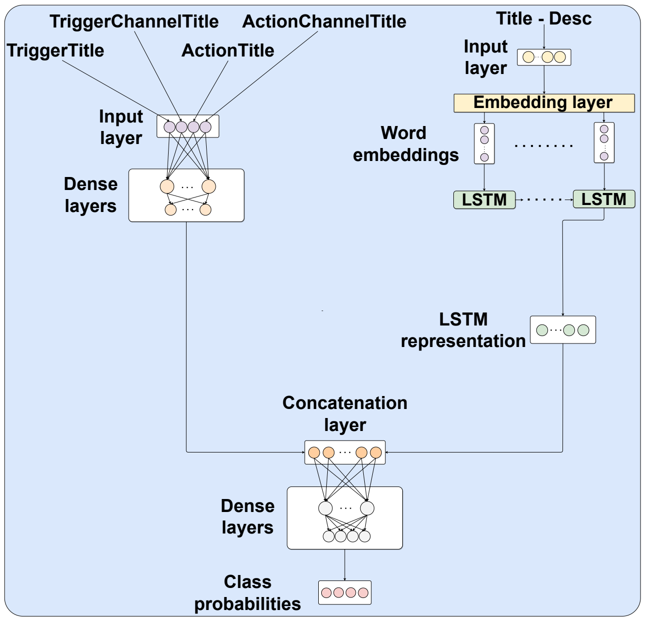

Fig. 7. The architecture of the LSTM-3c model.

## B. Classification by BERT

In the following, we present the classifiers developed with BERT by considering the same three combinations of features, but all in a textual form. BERT is based on the concept of Transfer Learning, namely a Machine Learning technique in which a model exploits knowledge gained from a problem to improve its performance on a related one. Unlike directional models (e.g., the LSTM model), which read the textual input sequentially (from left to right or vice versa), BERT, as a contextual model, captures the relationships between words in a bidirectional manner. In this way, it can learn the context of a word based on everything around it.

We train BERT by freezing all the pre-trained layers, and a layer of untrained neurons is added to the top of the architecture. Thus, during the training phase, only the additional classification layer is trained on the dataset. The classifier has been implemented using the `BertForSequenceClassification` class of the TRANSFORMERS Python library, which corresponds to the BERT model with a single linear layer added at the top for classification. Among the different available pre-trained BERT models, we use the “bert-base-uncased”, which represents the “base” version with 12 transformer blocks, 768 hidden units, 12 self-attention heads, and considers lowercase letters.

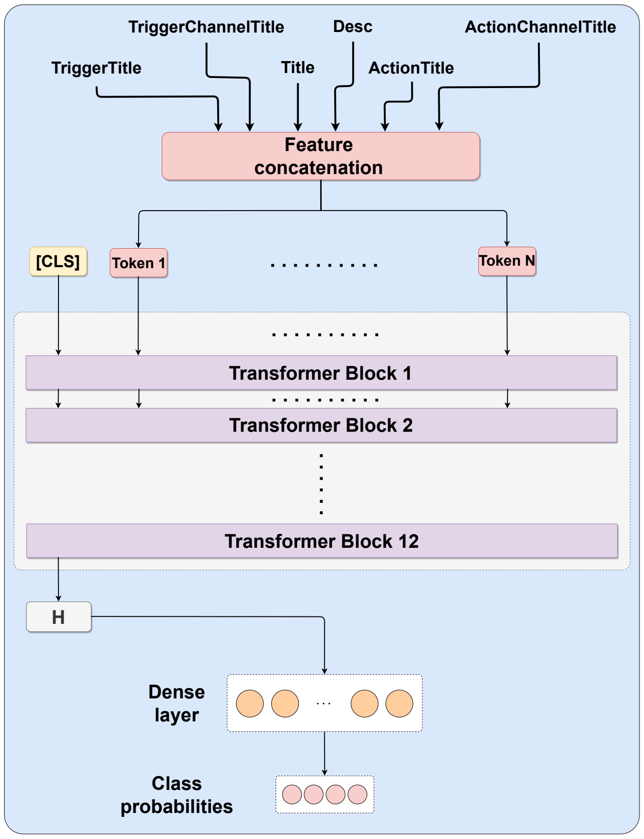

Fig. 8. Architecture of the BERT-based model considering all applet’s features (BERT-3c).

Fig. 8 shows the architecture of the implemented BERT-based model with all applet’s features in input. In particular, the input is a sequence of tokens where a special classification token, denoted with *[CLS]*, is placed at the beginning. All tokens are first embedded and then processed in the following blocks. Each block applies self-attention [48] and provides the output to a feed-forward neural network. Finally, the representation corresponding to the token [CLS] ($H$) is given in input to the Dense Layer added at the top of the architecture, which is responsible for classifying the input applet.

The models are trained by considering the following hyperparameters: 32 as batch size, $2e-5$ as learning rate, 2 epochs, and $1e-8$ as epsilon. This configuration is used with a maximum sentence length that changes based on the combination of considered features. In particular, for the first combination of features, we implement a model named *BERT-1c* with a maximum length of 50, as done in Section VI-A1. For the second combination of features, we implement a model named *BERT-2c* with the maximum length of embeddings set to 20, which corresponds to the longer sentence (as shown in Fig. 9). With this choice, the semantic meaning of the longer sentences is better captured, but without excessively affecting the performances and the training quality of the model. Furthermore, it is also worth noting that the number of services provided by the IFTTT platform is constantly growing. Thus, the considered length allows us to correctly represent any new trigger/action characterized by many characters, that might be added in the future.

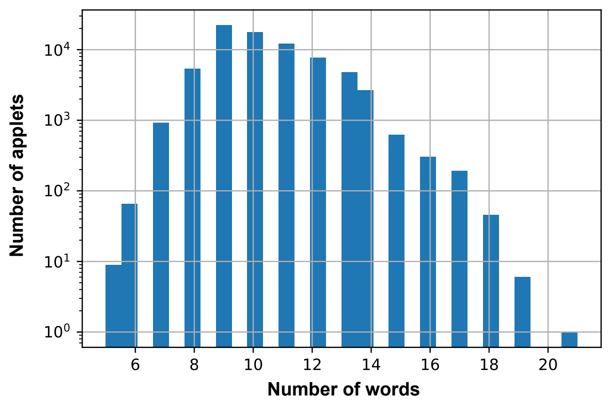

Fig. 9. Length distribution (in logarithmic scale) of the sentences obtained concatenating the discrete features.

Concerning the combination that considers all the features, we implement a model, named *BERT-3c*, with the maximum length of embeddings set to 70, which allows us to fully consider the discrete features (since they are inserted at the beginning of each sentence and are certainly correct as provided by the IFTTT platform) and in addition, in whole or in part, the title and the description, which can have variable length and potentially a more or less distorted meaning.

## *C. Training with Imbalanced Datasets*

The constructed dataset of IFTTT applets is "imbalanced" since the number of observations it contains is not the same for all classes. Imbalanced datasets are an important challenge in supervised classification [49]. In fact, a model trained on imbalanced data tends to classify the input instances with the class for which the most observations are available, namely the *majority class*. Different techniques can be used to alleviate the imbalanced dataset problem. In this work, we use "penalized" models, which try to penalize misclassifications related to minority classes more than those related to the majority class.

In general, the training objective of a model is the minimization of a loss function. Usually, every class in the loss function has the same *weight*, i.e., 1. Instead, in a penalized model these weights are altered for improving the accuracy of minority classes. In particular, it uses a *weighted loss function* that associates different weights to each class according to the number of class samples in the dataset. For the considered dataset, the minimum weight is associated with class 0, as it contains the largest number of samples.

We use the *categorical crossentropy* as a loss function for our multiclass classification task. To calculate the weight to be associated with each class, the `compute_class_weight` method of the SCIKIT-LEARN Python library is used. In particular, having indicated the string 'BALANCED' as a parameter, each weight is calculated as:

$$weight[C_i] = \frac{\#samples}{\#classes \cdot \#samples[C_i]}$$

where

- $\#samples$: represents the total number of instances considered for training;
- $\#classes$: represents the total number of considered classes;
- $\#samples[C_i]$: represents the total number of instances of class $C_i$ considered for training.

The loss for a sample $x$ of class $C_i$ is calculated as:

$$loss(x, C_i) = weight[C_i] \cdot \Big( -x[C_i] + \log\big(\textstyle\sum_j \exp(x[j])\big)\Big)$$

Imbalanced datasets could also introduce biased estimations or overfitting in favor of the majority class when using k-fold cross validation to evaluate the performances of the models. To alleviate this problem, we use the *stratified k-fold cross validation* technique, which performs stratified rather than random sampling. Stratified sampling is a probabilistic sampling procedure whereby reference data are divided into subsets that are as homogeneous as possible to the variable whose value is to be estimated. It ensures that the folds of the data have a uniformly representative sampling of the target attribute.

We use the stratified $k$-fold cross validation for fine tuning hyperparameters. In particular, we employ it for the optimization of the weights to be adopted in the loss function previously introduced. We set *k* to 4 in this work.

## VII. EXPERIMENTAL EVALUATION

We conduct a series of experiments to analyze the performances of the implemented models. This section illustrates the experimental setup, the adopted metrics, and the experimental results. The latter are compared with those obtained by a solution evaluating the risk of IFTTT applets through the information flow analysis [11].

## *A. Evaluation Setup*

Each of the previously described models is trained on the set of 76,741 applets obtained by the ensemble approach, whereas the 2,473 manually labeled applets are used as a test set (named TS_2k). Moreover, since the training set is labeled through the application of semi-supervised techniques to the TS_2k applets, we make a further evaluation on 500 applets randomly chosen among the applets not yet labeled (named TS_500). In this way, we validate the quality of the proposed methodology, i.e., the performances of the classification models to properly discriminate applets' damage, and the quality of the labels provided by the ensemble approach.

Once the training and test sets are selected, as said above, we apply a weighted loss function to solve the imbalanced dataset problem, obtaining the weights through the stratified k-fold cross validation. As a new validation set is created at each iteration, it is not necessary to extract one from the training set. Therefore, once the weights are obtained, each model is trained on all 76,741 applets.

## B. Evaluation Metrics

The performances of each classifier are evaluated by considering the following metrics:

- *Accuracy*: is the ratio between the number of instances correctly classified and the total number of considered instances;
- *Precision*($C_i$): is the ratio between the number of instances of class $C_i$ correctly classified and the total number of instances to which the classifier associates class $C_i$;
- *Recall*($C_i$): is the ratio between the number of instances of class $C_i$ correctly classified and the total number of instances of the test set labeled with class $C_i$;
- *F1-score*($C_i$): is the harmonic mean of Precision and Recall;

where $C_i$ is one of the four classes of damage. We also compute the average of the per-class metrics, weighted by the number of samples for each class in the test set (*WAvg*).

## C. Results and Discussion

*1) TS_2k Evaluation:* Tables II-V report the values of Accuracy, Precision, Recall, and F1-score obtained by the different models on the dataset TS_2k. We can observe that the worst results are achieved by the models trained considering discrete features only (i.e., NN-2c and BERT-2c), whereas the best results are achieved by the BERT model trained on all features (BERT-3c), which obtained 88% for all the metrics.

LSTM-3c and BERT-1c are the models that obtained results closer to those of BERT-3c. However, it is worth noting that the LSTM-3c model achieves a Precision for class 0 (72%) lower than BERT-3c. This means that it classifies many applets that provide damage as Innocuous, and this is particularly dangerous for users. At the same time, this model obtains higher values for Recall of classes 0, 1, and 2, but a very low value for class 3 (22%). This means that it rarely classifies an applet with class 3, as also highlighted by the high Recall values of the other classes. In addition, since the lowest Precision values are obtained by LSTM-3c for classes 0 and 1 (72% and 77%, respectively), we infer that the model erroneously classifies most of class 3 applets with one of these two classes.

The BERT-1c model is characterized by performances very similar to those of BERT-3c, achieving slightly higher Recall (for classes 1, 2, and 3) and Precision (for classes 0 and 3) values, but lower values for Accuracy (86%) and weighted average results. Concerning the F1-score metric, the BERT-1c model shows a slightly higher value only for class 3, confirming that the BERT-3c model performs better on average. Since the BERT-1c model classifies the applets by only considering their title and description, the performances of the model strongly depend on the semantic consistency of the applets' title and description contained in the training set. This information is specified by the user when creating the applet, and it might happen that it is *inconsistent* with the applet's behavior. In these cases, the BERT-3c model can exploit the discrete features (trigger, action, and the corresponding channels) populated by the IFTTT platform.

By focusing on the results per class, we can observe that for some models the applets of classes 1 and 3 are the most difficult to identify. For class 1 applets, the reason is that they could be erroneously classified with class 2 or 3 due to their slight differences in the context of use. As an example, the applet "`Any new photo by me uploaded in a specific Google Drive folder, publish it on Twitter`" should be classified as class 1 because a user could share an embarrassing photo unintentionally, whereas the applet "`Any new photo uploaded by anyone in a specific Google Drive folder, publish it on Twitter`" should be classified as class 3, because the user's privacy may be compromised as it is not possible to know who will upload the photo that will appear on his/her Twitter profile. On the other hand, the applet "`New tweet by me with a specific hashtag, turn off lights`" could be used by a user to turn off lights with a goodnight tweet, but it might be triggered also in other situations causing the lights to go off unintentionally, and consequently, the applet should be classified as class 1. Instead, the applet "`New tweet by anyone in the area with a specific tag, turn on lights`" allows a user to know if there are people that published a tweet in the zone, but its behavior might be compromised by a third party causing damage to the lights, and consequently, the applet should be classified as class 2. The differences between these applets are hard to grasp, and justify the low performances of the models in the discrimination of class 1 applets with respect to other classes.

Regarding class 3 applets, as said above, they are difficult to classify by the LSTM-3c model and by the models using only the information provided by the IFTTT platform (the discrete features on trigger and action). This might be due to the fact that the dataset considered for this work does not contain information about the *Fields* of the applets. To understand this aspect, let us consider the following applet: "`Record your daily Fitbit activity in a Google Spreadsheet`". In this case, the IFTTT platform upon creating such an applet requires the user to specify the spreadsheet to be involved in the automation, which is a field of the applet. Depending on the selected spreadsheet, the applet may or may not compromise the user's privacy. This is because, if the user specifies a private spreadsheet, which only s/he can access, then such an applet would not pose any risk. Conversely, if the user specifies a shared spreadsheet, then his/her privacy may be compromised as private information may be leaked to other people.

*2) TS_500 Evaluation:* To verify if the considered models generalize well on new applets, we perform an additional evaluation by considering a new test set obtained through the random selection of 500 applets among those discarded by the ensemble approach (i.e., the applets that obtained conflicting predictions by the three semi-supervised techniques). The result of the manual classification on these new applets is the TS_500 dataset containing: 264 applets belonging to class 0, 161 to class 1, 28 to class 2, and 47 to class 3.

Table VI reports the values of Accuracy and F1-score obtained by the classification models on TS_500. We can observe that all models perform worse than the evaluation on TS_2k,

TABLE II
ACCURACY OF THE CONSIDERED MODELS ON TS_2K

| Model | Accuracy (%) |
|---|---|
| LSTM-1c | 80 |
| BERT-1c | 86 |
| NN-2c | 31 |
| BERT-2c | 72 |
| LSTM-3c | 79 |
| BERT-3c | **88** |

TABLE III
PRECISION OF THE CONSIDERED MODELS ON TS_2K

| | Precision (%) | | | | |
|---|---|---|---|---|---|
| Model | class 0 | class 1 | class 2 | class 3 | WAvg |
| LSTM-1c | 73 | 80 | 86 | 87 | 80 |
| BERT-1c | **90** | 80 | 85 | 88 | 86 |
| NN-2c | 47 | 26 | 31 | 25 | 35 |
| BERT-2c | 73 | 66 | 80 | 69 | 72 |
| LSTM-3c | 72 | 77 | **91** | **93** | 81 |
| BERT-3c | 89 | **87** | 88 | 87 | **88** |

TABLE IV
RECALL OF THE CONSIDERED MODELS ON TS_2K

| | Recall (%) | | | | |
|---|---|---|---|---|---|
| Model | class 0 | class 1 | class 2 | class 3 | WAvg |
| LSTM-1c | 81 | 62 | 90 | 86 | 80 |
| BERT-1c | 77 | 82 | 98 | **95** | 86 |
| NN-2c | 24 | 32 | 48 | 28 | 31 |
| BERT-2c | 54 | 78 | 94 | 78 | 72 |
| LSTM-3c | **90** | **93** | **99** | 22 | 79 |
| BERT-3c | 86 | 79 | 97 | 91 | **88** |

TABLE V
F1-SCORE OF THE CONSIDERED MODELS ON TS_2K

| | F1-score (%) | | | | |
|---|---|---|---|---|---|
| Model | class 0 | class 1 | class 2 | class 3 | WAvg |
| LSTM-1c | 77 | 70 | 88 | 86 | 80 |
| BERT-1c | 83 | 81 | 91 | **91** | 86 |
| NN-2c | 32 | 28 | 37 | 27 | 31 |
| BERT-2c | 62 | 71 | 87 | 73 | 71 |
| LSTM-3c | 80 | **84** | **95** | 36 | 75 |
| BERT-3c | **87** | 83 | 92 | 89 | **88** |

except for BERT-3c, which increases its Accuracy (F1-score, resp.) from 88% to 91% (92%, resp.). In the following, we provide a detailed analysis of the results achieved by BERT-3c. For this model, we also analyze whether semantically consistent titles and descriptions influence its performances. To this end, we manually verify the consistency of the applet's titles and descriptions with respect to the applet's behavior. This process leads to the identification of 75 applets whose titles and descriptions do not describe the goal of the automation. The dataset without these applets (named TS_425) consists of 209 class 0 applets, 147 of class 1, 27 of class 2, and 42 of class 3.

TABLE VI
ACCURACY AND F1-SCORE ACHIEVED BY THE CONSIDERED MODELS ON TS_500

| Model | Accuracy (%) | WAvg F1-score (%) |
|---|---|---|
| LSTM-1c | 60 | 57 |
| BERT-1c | 71 | 71 |
| NN-2c | 29 | 29 |
| BERT-2c | 63 | 61 |
| LSTM-3c | 75 | 73 |
| BERT-3c | **91** | **92** |

Table VII reports the values of the metrics obtained by BERT-3c on TS_500 and TS_425. The results are better than those obtained on TS_2k in almost all classes, as also highlighted by F1-score. In fact, BERT-3c correctly discriminates class 1 applets, differently from what happened previously. However, the results for class 2 applets are considerably worsened.

Tables VIII and IX show the confusion matrices obtained from the classification results on TS_500 and TS_425. We can observe that the model's performances are slightly better when only *consistent* applets are classified, confirming that this applet's property allows the model to better discriminate among classes. By analyzing the Recall, we can observe that BERT-3c correctly classifies all class 2 and class 3 applets, while it makes some mistakes on the other two classes. In fact, with this evaluation, we discover another ambiguity in discriminating between class 2 and class 0 applets. As an example, the applet with description "`Email a map of where I parked`" might be classified as class 2 because a user could share his/her car position unintentionally, and a third party might exploit this information to cause damage. However, as the ActionTitle of this applet is "`Send me an email`", it should be classified as class 0 because the information remains private. Since the number of class 0 applets is much higher than those of class 2, it happens that these misclassifications for class 0 applets are relatively frequent with respect to class 2 applets, leading to a very low precision value for class 2.

## D. Comparative Evaluation

To further assess the validity of the proposed approach, we compare the performances of the BERT-3c model with those of a baseline system implementing an information flow analysis similar to the one proposed in [11]. The latter exploits a secrecy lattice to identify possible violations caused by an IFTTT applet. The lattice of the baseline system introduces two levels of restriction, namely public and private, asserting that a secrecy violation occurs when the information flows from the private level toward the public one. Considering this type of approach, we devise a methodology following such a principle for classifying an applet through the analysis of its channels, namely *Information Flow Classifier* (IFC). The classification labels of IFC are:

- *Harmless*: indicates that the applet contains no elements that may lead to violations;
- *Harmful*: indicates that the applet contains elements that may lead to violations.

To implement the baseline system, we first extract all the channels of the IFTTT platform from the constructed dataset. Each channel belongs to one of the following categories:

- Smart Objects;

TABLE VII
PERFORMANCES OF BERT-3C ON TS_500 AND ITS SUBSET OF "CONSISTENT" APPLETS TS_425

| | Accuracy (%) | | Precision (%) | | Recall (%) | | F1-score (%) | |
|---|---|---|---|---|---|---|---|---|
| | TS_500 | TS_425 | TS_500 | TS_425 | TS_500 | TS_425 | TS_500 | TS_425 |
| Class 0 | | | 97 | 98 | 88 | 89 | 92 | 94 |
| Class 1 | | | 94 | 99 | 94 | 95 | 94 | 97 |
| Class 2 | | | 54 | 59 | 100 | 100 | 70 | 74 |
| Class 3 | | | 90 | 89 | 96 | 100 | 93 | 94 |
| WAvg | 91 | 93 | 93 | 95 | 91 | 93 | 92 | 93 |

TABLE VIII
CONFUSION MATRIX OBTAINED FOR TS_500

| Target Class \ Output Class | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 232 97% | 8 5% | 22 42% | 2 4% |
| 1 | 6 3% | 151 94% | 1 2% | 3 6% |
| 2 | 0 0% | 0 0% | 28 54% | 0 0% |
| 3 | 0 0% | 1 1% | 1 2% | 45 90% |

TABLE IX
CONFUSION MATRIX OBTAINED FOR TS_425

| Target Class \ Output Class | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 187 98% | 2 1% | 18 39% | 2 4% |
| 1 | 4 2% | 139 99% | 1 2% | 3 6% |
| 2 | 0 0% | 0 0% | 27 59% | 0 0% |
| 3 | 0 0% | 0 0% | 0 0% | 42 89% |

TABLE X
A COMPARISON OF THE RESULTS OBTAINED BY IFC AND BINARYBERT-3C FOR TS_2K

| | Accuracy (%) | Precision (%) | | Recall (%) | | F1-score (%) | |
|---|---|---|---|---|---|---|---|
| Methodology | | class HL | class HF | class HL | class HF | class HL | class HF |
| IFC | 46 | 41 | 70 | 87 | 20 | 56 | 32 |
| BinaryBERT-3c | 92 | 89 | 93 | 90 | 93 | 89 | 93 |

- Cloud Services;
- Social Networks.

We assign to each channel one of the following labels:

- *Private*: refers to channels that by default tend to privatize the information they manage, e.g., Google Drive by default allows users to privately store their files;
- *Public*: refers to channels that by default tend to publicly share the information they manage, e.g., Facebook by default allows other people to view the content that a user uploads on the platform.

The channels belonging to the "Smart Objects" or "Cloud Services" categories are categorized as operating in private contexts, while the "Social Networks" channels are categorized as operating in public contexts.

The process of applet labeling is performed as follows:

- If the trigger channel is labeled as "Private" and the action channel as "Public", then the applet is labeled as "Harmful";
- in all other cases the applet is labeled as "Harmless".

The reasoning behind such a choice is based on the assumption made in [11], where the authors state that "*it is safe to allow information to flow from a lower (public or trusted) label to a higher (private or untrusted) label, but not the other way around*". Here, the terms higher and lower refer to the nodes of the secrecy lattice.

Since this methodology allows us to classify an applet as Harmful or not, it is necessary to modify the labels of our dataset in terms of binary classification. In particular, the previous applets of classes 1, 2, and 3 are modified with the new class *HF*, which corresponds to "Harmful", whereas the class 0 applets are labeled with the new class *HL*, corresponding to "Harmless". Thus, to compare the two classifiers, we train the BERT-3c model on the dataset with binary labels. We will refer to this model as BinaryBERT-3c.

For the evaluation phase, we use the test set TS_2k obtaining the results reported in Table X. We can observe that IFC is not capable of accurately identify Harmful applets as highlighted by the Recall value of class *HF* (20%) and the Precision value of class *HL* (41%). This is due to the prominent lack of contextual information available when IFC classifies an applet. In particular, a static approach that considers only the trigger and action channels provides a too high generalization of the applet's behavior. In fact, as highlighted by the literature, the classification of an applet with respect to different classes of damage can be strongly derived from the analysis of its semantic components [10], [12]. As an example, if we consider the classification with IFC of the applet introduced in Section VII-C, the trigger channel "Fitbit" and the action channel "Google Drive" would be labeled as *Private*, and the applet is classified as Harmless. Conversely, BinaryBERT-3c exploits NLP techniques that allow to extract the semantic meaning of the descriptions and understand the applet's action context, which help to disambiguate the rules that are difficult to classify by analyzing only the trigger and the action. In fact, as shown in Table X, BinaryBERT-3c is capable of correctly classifying 92% of applets, as highlighted by the Accuracy value. In addition, high values for Precision, Recall, and F1-score in both classes provide a further hint about the ability of the model to classify them.

## VIII. Conclusion and Future Work

In this article, we have presented an approach to automatically identify individual ECA rules that are potentially dangerous for the security of the smart environment and for the privacy of the user. The approach exploits the capability of NLP models to semantically analyze the information of the rules, fostering an appropriate identification of the risks associated with them. The effectiveness of the proposed approach has been successfully demonstrated on the IFTTT platform, by considering different ANN and BERT-based models trained on a dataset of 76,741 applets. The labeling process of the dataset exploited semi-supervised learning techniques and an ensemble method. The evaluation of the results highlighted that the BERT-3c classifier, fine-tuned considering all rule's features, achieved the highest Precision and Recall values, scoring on average 88% for the TS_2k test set and 93% for the TS_425 test set. BERT-3c has also been compared on a binary classification task with an approach based on information flow analysis. This evaluation proved that our approach best fits the task of classifying harmful ECA rules since it is able to extract the semantic and contextual meaning of the rules.

In the future, we would like to further improve the performances of the classification models by enhancing the quality of the training set. In fact, as highlighted during the evaluation, several applets are characterized by descriptions semantically inconsistent with the actual applet behavior. Thus, we intend to introduce a pre-processing phase exploiting language generation models to obtain training sets with consistent descriptions. Other important rule features that could be considered are the *Fields* (see Section IV). To date, such information is not available in any dataset, since their values are assigned by the user when the applet is activated. In the future, we would like to train the models on a set of instantiated applets. For achieving such a goal, we should recruit a conspicuous number of participants to provide examples of how they would set up an ECA rule. Finally, we are also planning to expand our model with the capabilities to evaluate applets' risks when they are activated sequentially, generating a cause-effect relation, i.e., the action of a rule can trigger another one. In fact, some rules might be harmless if considered alone, while their interaction may give rise to some type of damage to users [3].

## Appendix A

Supplementary material related to this article can be found online at https://github.com/empathy-ws/Harmful-ECA-rules-classifiers. The repository provides the source code and datasets used in the experimental sections of the paper, as well as detailed notebooks showing the use of the provided models.

## References

[1] A. Krishna, M. Le Pallec, R. Mateescu, and G. Salaün, "Design and deployment of expressive and correct web of things applications," *ACM Trans. Internet Technol.*, vol. 3, no. 1, pp. 1–30, 2021.

[2] B. A. Johnsson and B. Magnusson, "Towards end-user development of graphical user interfaces for internet of things," *Future Gener. Comput. Syst.*, vol. 107, pp. 670–680, 2020.

[3] Q. Wang, P. Datta, W. Yang, S. Liu, A. Bates, and C. A. Gunter, "Charting the attack surface of trigger-action IoT platforms," in *Proc. Conf. on Comp. and Comm. Sec.* ACM, 2019, pp. 1439–1453.

[4] G. Desolda, C. Ardito, and M. Matera, "Empowering end users to customize their smart environments: model, composition paradigms, and domain-specific tools," *ACM Trans. Comput. Hum. Interact.*, vol. 24, no. 2, pp. 1–52, 2017.

[5] G. Ghiani, M. Manca, F. Paternò, and C. Santoro, "Personalization of context-dependent applications through trigger-action rules," *ACM Trans. Comput. Hum. Interact.*, vol. 24, no. 2, pp. 1–33, 2017.

[6] J. Cano, E. Rutten, G. Delaval, Y. Benazzouz, and L. Gurgen, "ECA rules for IoT environment: A case study in safe design," in *Proc. IEEE 8th Int. Conf. on Self-Adaptive and Self-Organizing Systems Workshops*. IEEE, 2014, pp. 116–121.

[7] V. Zhao, L. Zhang, B. Wang, S. Lu, and B. Ur, "Visualizing differences to improve end-user understanding of trigger-action programs," in *Extended Abstracts of the 2020 CHI Conf. on Hum. Factors Comput. Syst.* ACM, 2020, pp. 1–10.

[8] F. Corno, L. De Russis, and A. M. Roffarello, "A high-level semantic approach to end-user development in the internet of things," *Int. J. Hum. Comput.*, vol. 125, pp. 41–54, 2019.

[9] B. Breve, G. Desolda, V. Deufemia, F. Greco, and M. Matera, "An end-user development approach to secure smart environments," in *Proc. 8th Int. Symp. on End-User Development*. Berlin, Heidelberg: Springer-Verlag, 2021, p. 36–52.

[10] C. Cobb, M. Surbatovich, A. Kawakami, M. Sharif, L. Bauer, A. Das, and L. Jia, "How risky are real users' IFTTT applets?" in *Proc. 16th USENIX Conf. on Usable Privacy and Security*. USENIX Association, 2020, pp. 505–529.

[11] M. Surbatovich, J. Aljuraidan, L. Bauer, A. Das, and L. Jia, "Some recipes can do more than spoil your appetite: Analyzing the security and privacy risks of IFTTT recipes," in *Proc. 26th Int. Conf. on World Wide Web*. ACM, 2017, p. 1501–1510.

[12] M. Saeidi, M. Calvert, A. W. Au, A. Sarma, and R. B. Bobba, "If this context then that concern: Exploring users' concerns with IFTTT applets," *Proc. on Privacy Enhancing Technologies*, vol. 2022, pp. 166 – 186, 2021.

[13] S. Zheng, N. Apthorpe, M. Chetty, and N. Feamster, "User perceptions of smart home IoT privacy," *Proc. ACM on Human-Computer Interaction*, vol. 2, pp. 1–20, 2018.

[14] F. Paci, D. Bianchin, E. Quintarelli, and N. Zannone, "IFTTT privacy checker," in *Emerging Technologies for Authorization and Authentication*, A. Saracino and P. Mori, Eds. Cham: Springer Int. Publishing, 2020, pp. 90–107.

[15] M. McCall, F. H. Shezan, A. Bichhawat, C. Cobb, L. Jia, Y. Tian, C. Grace, and M. Yang, "SAFETAP: An efficient incremental analyzer for trigger-action programs," Carnegie Mellon University, Tech. Rep., 2021.

[16] D. Xiao, Q. Wang, M. Cai, Z. Zhu, and W. Zhao, "A3ID: An automatic and interpretable implicit interference detection method for smart home via knowledge graph," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2197–2211, 2019.

[17] C. Liu, X. Chen, E. C. R. Shin, M. Chen, and D. X. Song, "Latent attention for if-then program synthesis," in *Proc. Annual Conf. on Neural Inf. Process. Syst.* NeurIPS, 2016, pp. 4574–4582.

[18] Y. Luo, L. Cheng, H. Hu, G. Peng, and D. Yao, "Context-rich privacy leakage analysis through inferring apps in smart home IoT," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2736–2750, 2020.

[19] X. Mi, F. Qian, Y. Zhang, and X. Wang, "An empirical characterization of IFTTT: ecosystem, usage, and performance," in *Proc. Internet Measurement Conference*. ACM, 2017, pp. 398–404.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. of the North American Chapter of the ACL: Human Language Technologies, Volume 1*, 2019, pp. 4171–4186.

[21] B. Breve, G. Cimino, and V. Deufemia, "Towards a classification model for identifying risky IFTTT applets," in *Proc. 2nd Int. Workshop on Empowering People in Dealing with Internet of Things Ecosystems co-*

*located with INTERACT'21*, ser. CEUR Workshop Proceedings, vol. 3053, 2021, pp. 33–37.

[22] M. Benerecetti, F. Giunchiglia, and L. Serafini, "Model checking multiagent systems," *Journal of Logic and Computation*, vol. 8, no. 3, pp. 401–423, 1998.

[23] L. d. Moura, B. Dutertre, and N. Shankar, "A tutorial on satisfiability modulo theories," in *Proc. Int. Conf. on Computer Aided Verification*. Springer, 2007, pp. 20–36.

[24] W. Ding and H. Hu, "On the safety of IoT device physical interaction control," in *Proc. ACM Conf. on Comp. and Comm. Sec.*, 2018, pp. 832–846.

[25] K.-H. Hsu, Y.-H. Chiang, and H.-C. Hsiao, "SafeChain: Securing trigger-action programming from attack chains," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 10, pp. 2607–2622, 2019.

[26] Q. Wang, W. U. Hassan, A. Bates, and C. Gunter, "Fear and logging in the internet of things," in *Proc. 25th Annual Network and Distributed System Security Symposium*. National Science Foundation, 2018.

[27] R. Xu, Q. Zeng, L. Zhu, H. Chi, X. Du, and M. Guizani, "Privacy leakage in smart homes and its mitigation: IFTTT as a case study," *IEEE Access*, vol. 7, pp. 63 457–63 471, 2019.

[28] I. Bastys, M. Balliu, and A. Sabelfeld, "If this then what? Controlling flows in IoT apps," in *Proc. ACM Conf. on Comp. and Comm. Sec.* ACM, 2018, p. 1102–1119.

[29] Y.-H. Chiang, H.-C. Hsiao, C.-M. Yu, and T. H.-J. Kim, "On the privacy risks of compromised trigger-action platforms," in *Proc. 25th European Symposium on Research in Computer Security*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 251–271.

[30] Y. Chen, A. R. Chowdhury, R. Wang, A. Sabelfeld, R. Chatterjee, and E. Fernandes, "Data privacy in trigger-action systems," in *Proc. IEEE Symposium on Security and Privacy*. IEEE, 2021, pp. 501–518.

[31] J. Fan, Y. He, B. Tang, Q. Li, and R. Sandhu, "Ruledger: Ensuring execution integrity in trigger-action IoT platforms," in *Proc. IEEE Conf. on Comp. Comm.* IEEE, 2021, pp. 1–10.

[32] Z. B. Celik, G. Tan, and P. D. McDaniel, "IoTGuard: Dynamic enforcement of security and safety policy in commodity IoT," in *Proc. Annual Network and Distr. Syst. Sec. Symp.*, 2019.

[33] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "FlowFence: Practical data protection for emerging IoT application frameworks," in *Proc. 25th USENIX Security Symposium*. USENIX Association, 2016, pp. 531–548.

[34] Z. B. Celik, L. Babun, A. K. Sikder, H. Aksu, G. Tan, P. McDaniel, and A. S. Uluagac, "Sensitive information tracking in commodity IoT," in *Proc. 27th USENIX Security Symposium*. USENIX Association, 2018, pp. 1687–1704.

[35] Z. B. Celik, P. McDaniel, and G. Tan, "Soteria: Automated IoT safety and security analysis," in *Proc. USENIX Annual Technical Conference*. USENIX Association, 2018, pp. 147–158.

[36] D. T. Nguyen, C. Song, Z. Qian, S. V. Krishnamurthy, E. J. Colbert, and P. McDaniel, "IoTSan: Fortifying the safety of IoT systems," in *Proc. 14th Int. Conf. on Emerging Networking EXperiments and Technologies*. ACM, 2018, pp. 191–203.

[37] J. Wang, S. Hao, R. Wen, B. Zhang, L. Zhang, H. Hu, and R. Lu, "IoT-Praetor: Undesired behaviors detection for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 927–940, 2021.

[38] M. J. Jozani, É. Marchand, and A. Parsian, "On estimation with weighted balanced-type loss function," *Statistics & Probability Letters*, vol. 76, no. 8, pp. 773–780, 2006.

[39] N. Diamantidis, D. Karlis, and E. A. Giakoumakis, "Unsupervised stratification of cross-validation for accuracy estimation," *Artif. Intell.*, vol. 116, no. 1-2, pp. 1–16, 2000.

[40] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015.

[41] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing*. ACL, 2019, pp. 3982–3992.

[42] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *Proc. 33rd Annual Meeting of the Association for Computational Linguistics*. ACM, 1995, pp. 189–196.

[43] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Carnegie Mellon University, Tech. Rep. CMU-CALD-02-107, 2002.

[44] D. Croce, G. Castellucci, and R. Basili, "GAN-BERT: Generative adversarial learning for robust text classification with a bunch of labeled examples," in *Proc. 58th Annual Meeting of the Association for Comp. Ling.* ACL, 2020, pp. 2114–2119.

[45] R. Wang, R. Ridley, W. Qu, X. Dai *et al.*, "A novel reasoning mechanism for multi-label text classification," *Information Processing & Management*, vol. 58, no. 2, p. 102441, 2021.

[46] S. Ghannay, B. Favre, Y. Esteve, and N. Camelin, "Word embedding evaluation and combination," in *Proc. 10th Int. Conf. on Language Resources and Evaluation*. ELRA, 2016, pp. 300–305.

[47] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. on Empirical Methods in Natural Language Processing*. ACL, 2014.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[49] L. Wang, M. Han, X. Li, N. Zhang, and H. Cheng, "Review of classification methods on unbalanced data sets," *IEEE Access*, vol. 9, pp. 64 606–64 628, 2021.

**Bernardo Breve** received his B.Sc. and M.Sc. degrees in Computer Science from the University of Salerno in 2016 and 2019, respectively. He is a Ph.D. researcher at the Department of Computer Science of the University of Salerno, with a scholarship founded by the PRIN 2017 "EMPATHY: Empowering People in deAling with internet of THings ecosYstems" project, granted by the Italian Ministry of University and Research (MUR). His research interests include natural language processing, data science, and human-computer interaction, with an emphasis on usable security and privacy for end-users.

**Gaetano Cimino** received his B.Sc. and M.Sc. degrees in Computer Science from the University of Salerno in 2019 and 2021, respectively. He is a Ph.D. researcher at the Department of Computer Science of the University of Salerno, with a scholarship founded by the "Department of Excellence 2018–2022" (Law 232/2016) project, granted by the Italian Ministry of University and Research (MUR). He regularly serves as a reviewer for international conferences and journals. As part of his Ph.D. studies, he focuses on Natural Language Processing, Explainable Artificial Intelligence, and Applied Machine Learning.

**Vincenzo Deufemia** received the Laurea Degree (cum laude), and the PhD degree in computer science from the University of Salerno, in 1999 and 2003, respectively. He has been a visiting researcher with the Laboratory of Advanced enterprise Information Management Systems (AeIMS) of University of Western Sydney, Australia, in 2013. He is currently an associate professor with the Department of Computer Science, University of Salerno. He serves as a reviewer for several international journals and has been a member of international conference committees. He is a member of various associations, including IEEE and ACM. Recently, he has focused on topics related to natural language processing, usable security in IoT, and data science.