Learning Personalized and Context-Aware Violation Detection Rules in Trigger-Action Apps

Mahsa Saeidi[1], Sai Sree Laya Chukkapalli[2], Anita Sarma[3] and Rakesh B. Bobba[3]
[1]*University of Tehran, Tehran, Iran*
[2]*IBM Research, New York, U.S.A.*
[3]*Oregon State University, Oregon, U.S.A.*

Keywords: Privacy, Trigger-Action, Violation Detection Rules.

Abstract: Trigger-action apps are being increasingly used by end users to connect smart devices and online services to create new functionality. However, these apps can cause undesirable implicit information flows (secrecy violation) or lead to unintended accesses (integrity violation) depending on the usage context. Existing solutions designed to address such risks rely on predefined rules to control and mitigate such implicit information flows or unintended accesses. However, defining such rules is difficult for end users. In this work, we propose a learning-based approach to learn rules that flag violating situations based on the usage context. We also propose a set of reduction steps to reduce the complexity of the learned rules. We are able to achieve a good F1-measure in predicting both secrecy (0.91) and integrity (0.75) violations and achieve 77% and 74% complexity reduction while maintaining 88% and 97% of the original performance of the secrecy and integrity violation prediction, respectively.

1 INTRODUCTION

Trigger-action programming frameworks such as IFTTT (IFTTT, 2019), Microsoft Flow (Microsoft Corporation, 2019), and Zapeir (Zapier Inc., 2019) are getting increasingly popular. These frameworks enable end users to connect different IoT devices and online services using simple conditional logic (*e.g., If-this-then-that* in IFTTT) to enable new and rich functionality. It has been reported that for IFTTT alone, 18 million registered users were running over 1 billion applets each month in 2020 (McLaughlin, 2021).

While these platforms provide convenience and ease of use, there are inherent security and privacy risks. These include unexpected or undesirable implicit information flows (Surbatovich et al., 2017; Cobb et al., 2020; Celik et al., 2018; Fernandes et al., 2016). For example, an applet that automatically uploads pictures taken by a motion-activated smart home security camera during the day may cause privacy violations when unexpected guests arrive or home owners return early.

Surbatovich *et al.* (Surbatovich et al., 2017) highlighted such risks in IFTTT applets using a lattice-based analysis framework. They used security labels and information flow analysis to identify *secrecy* (confidentiality) and *integrity* violations. Multiple prior works have used a lattice-based model to determine security and privacy violations in smart-home solutions because of the simplicity and ease of static information-flow rules (Bastys et al., 2018; Celik et al., 2019). However, recent work by Cobb *et al.* (Cobb et al., 2020) has shown that lattice-based models cannot capture security and privacy risks accurately due to lack of contextual information. They discussed how contextual information such as device location within the home can help identify the violations more accurately.

Contextual information alone may not suffice to accurately capture risks. We show that even with usage context considered, lattice-based evaluation, as discussed in (Surbatovich et al., 2017; Cobb et al., 2020), does not yield a unique decision on violations, highlighting the influence of individual preferences.

Therefore, customizing security and privacy evaluation for each individuals' personal preference is going to be a necessity, which is inline with prior work (Saeidi et al., 2020) that showed end-user perception of security and privacy risks varies significantly between individuals. While some existing frameworks allow such personalization (*e.g.,* Expat (Yahyazadeh et al., 2019), IoTGuard (Celik et al.,

Saeidi, M., Chukkapalli, S. S. L., Sarma, A. and Bobba, R. B.
Learning Personalized and Context-Aware Violation Detection Rules in Trigger-Action Apps.
DOI: 10.5220/0013528800003979
In *Proceedings of the 22nd International Conference on Security and Cryptography (SECRYPT 2025)*, pages 429-442
ISBN: 978-989-758-760-3; ISSN: 2184-7711

2019)), where end users define policy rules through specialised policy languages or graphical user interfaces, doing so is difficult for end users (Smetters and Good, 2009; Bauer et al., 2009). The already complex task of defining policy rules will be made much more difficult by end users having to consider different usage contexts for different applets.

To address this gap, we propose a framework for learning integrity and secrecy violation detection rules tailored to individual preferences. We frame the problem as a classification task that uncovers violation detection rules from a synthesized training dataset, which includes various usage scenarios of multiple applets created by combining different contextual information. Additionally, we use reduction techniques to minimize complexity and redundancy in the final learned rules. In summary, this work:

- Validates that integrity and secrecy violations depend on both the context and user preferences using a synthesized dataset of smart-home applets labeled considering contextual information.
- Shows that machine learning models can learn and generalize violation detection rules for a given user. Our Random Forest (RF) model achieved good F1-measure in predicting both secrecy (0.91) and integrity (0.75) violations.
- Proposes reduction techniques to decrease the complexity of the learned rules, achieving a 77% and 74% decrease in complexity while maintaining 88% and 97% of the original performance in predicting secrecy and integrity violations, respectively.

The paper is structured as follows: Section 2 covers the threat model and lattice-based approach. In Section 3, we outline our method for creating a labeled smart-home dataset. Section 4 discusses the limitations of lattice-based models for violation detection and the role of contextual information. Section 5 presents our framework for learning violation detection rules and reducing complexity, along with its evaluation on the dataset. We address limitations and future work in Section 6 and review related work in Section 7. Finally, we conclude in Section 8.

2 THREAT MODEL AND LATTICE-BASED APPROACH

Threat Model. In this paper, we focus on violations that arise when unauthorized users (*e.g.,* visitors or outsiders) trigger smart home applets to perform unintended or malicious actions or to access sensitive information through them. For example, a visitor could issue a voice command to disable security cameras, thus compromising home security, or exploit an online service linked to a smart device to expose sensitive information, such as daily routines of residents and their presence or absence at home.

Our paper categorizes the above situations as integrity and secrecy violations and discusses how to determine and detect them. Specifically, an *integrity violation* happens if an applet[1] allows unauthorized or unintended access to a smart-home device (*e.g.,* a smart camera) or a resource on an online service (*e.g.,* a Google spreadsheet). On the other hand, A *secrecy* violation happens when an applet enables the flow of private information to a public destination (*e.g.,* uploading footage from an indoor camera, considered private, to a shared Google folder, considered public).

Lattice-Based Approach. One of the simple and understandable approaches that can help to identify the violations in smart-home apps is the lattice-based model that was proposed by prior works (Cobb et al., 2020; Surbatovich et al., 2017). A lattice-based security model detects integrity (undesirable access) and secrecy violations (undesirable information flow) by employing *security labels* that form a lattice, or partial order, along with simple static rules based on information flow analysis (Denning, 1976). Figure 1 shows the security lattice used in (Surbatovich et al., 2017) to analyze information flows and identify integrity and secrecy violations. The arrows in the figure indicate the permissible information flows and authorized accesses. For instance, information flow is allowed from devices or services with *Public* label to those with *Private* label, but not vice versa (Left side of Figure 1). Similarly, devices or services labeled *Trusted* are allowed to access or control those with *Untrusted* label, but not vice versa (Right side of Figure 1).

In applying a lattice-based security model to smart-home applets, integrity and secrecy labels for triggers and actions are assigned by responding to two questions: *who can access* and *who can observe the trigger and action* (Surbatovich et al., 2017). For instance, when a smart device is located within a home, it means only a restricted group of people who are physically present in the home could either access the device to initiate a trigger or observe the action event if it occurs in the device. In this case, the security label of the trigger/action would be *restricted physical*.

To see the security rule application, consider an applet that adds one row to the user's Google spreadsheet when the front door gets unlocked. In this case, an integrity violation can occur if an unknown

[1] In this paper, app and applet are used interchangeably to refer to IFTTT like recipes connecting smart-home devices and online services.

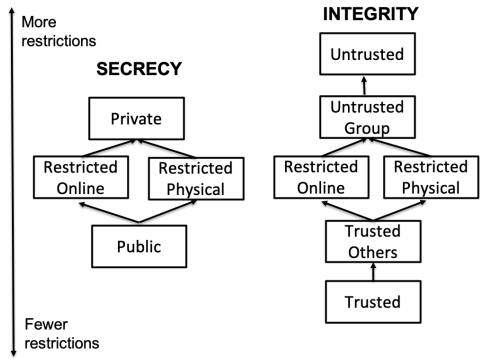

Figure 1: The integrity and secrecy (confidentiality) lattices used in (Surbatovich et al., 2017).

person (untrusted user) unlocks the door and causes the action of writing in the Google spreadsheet that is allowed only by the Google spreadsheet owner (a trusted user).

3 DATASET SYNTHESIS METHODOLOGY

To investigate how usage context affects the accuracy of lattice-based integrity and secrecy violation detection approaches, we need a trigger-action app dataset that includes contextual information. While smart-home datasets with smart sensor readings (Washington State University, 2019) and IFTTT app datasets (Surbatovich et al., 2017; Cobb et al., 2020) were available, these did not include contextual information. Therefore, we synthesized a dataset as described below[2].

Applet Selection: We chose our smart-home applets[3] from two IFTTT applet datasets used in previous work (Cobb et al., 2020; Saeidi et al., 2020). The first dataset includes 396 unique IFTTT applets, with 218 related to smart homes, while the second includes 49 unique smart-home applets. An ideal scenario would be to consider every applet that creates a unique trigger and action pair. However, this would make creating a labeled dataset very challenging as will become evident later. To reduce the dimensions of trigger/action events, we grouped triggers and actions into semantic categories proposed by (Cobb et al., 2020) with two additional subcategories (DS:A and DS:P) to differentiate between appliances and printers from other IoT devices within the home. We identified 39 unique pairs of trigger and action semantic categories across the two datasets and randomly sampled one applet from each pair of trigger and action semantic categories across both datasets (Cobb et al., 2020; Saeidi et al., 2020) to create a semantically representative applet dataset. Thus we picked 39 applets which represent the 39 unique pairs of semantic categories across the two datasets.

Contextual Factors: Next step in creating a suitable dataset for our analysis is identifying contextual factors that might influence or impact what might be a considered a violation in smart-home trigger-action apps. We identified a set of six contextual factors that can have a potential impact on integrity and secrecy violations by reviewing existing relevant work (Naeini et al., 2017; Lee and Kobsa, 2016; Lee and Kobsa, 2017; He et al., 2018; Saeidi et al., 2020; Cobb et al., 2021; Jin et al., 2022). These selected factors were also listed as the most mentioned contexts in a recent survey on contextual access controls in homes (He et al., 2021). However, we note that this set is not meant to be exhaustive but just a reasonable starting point to illustrate the proposed approach.

- **Trigger Location.** The area in the home where the trigger event can occur has security and privacy implications as it defines who can access the trigger (Saeidi et al., 2020). This contextual factor includes *Living Room* and *Kitchen* that are considered public areas, and *Bedroom* and *Bathroom* that are considered private areas.
- **Action Location.** The area in the home where the action event occurs also has security and privacy implications as it defines who is able to observe the action. Similar to *Trigger Location*, this factor includes public areas (Living Room, Kitchen) and private areas (Bedroom, Bathroom).
- **Time.** The time of day an end user uses the applet can impact security and privacy, as cameras may capture more sensitive footage at specific times. We consider three values for this factor: Morning, Afternoon, and Night.
- **Online Service Trustworthiness/Secrecy.** This factor includes two situations when the trigger/action occurs in online services. For *triggers*, online service trustworthiness indicates whether the trigger event is initiated by trusted or untrusted online services. For instance, in an applet that flashes a light when a new feed item matches the app rule, there might be security/privacy concerns if the RSS feed is from an untrusted website. For

[2]This dataset will be made available to the community.

[3]A smart-home applet is an applet in which either the trigger or action would happen in the smart-home environment.

actions, when these are reported/logged in online services, online service secrecy indicates whether the information available in online services is private or shared with others. For instance, in an applet that uploads a Facebook post when the user arrives home, there might be privacy concerns if the Facebook post is on a public page.

- **Trigger Device/Service.** The trigger device or service refers to the resource that initiates trigger events. It is essential to identify the trigger device or service to determine who has access to it and what kind of information may be exposed. For instance, if the trigger is a voice command activated through a voice assistant like Alexa, anyone in close proximity to the device can initiate the command. To identify the device or online service used by the applet, we utilized the semantic categories proposed by (Cobb et al., 2020) (*e.g.,* Semantic label "V" for triggers that are "voice" activated based on verbal interactions with a smart assistant such as, Alexa, Siri).
- **Action Device/Service.** This factor refers to the resource where the action events take place. The type of device or service, along with its features, can have an impact on privacy and security implications. For instance, having access to a light would be less sensitive than having access to a security camera. Similar to *Trigger Device/service* we used the semantic labels proposed by (Cobb et al., 2020) to identify the type of device and service for action event.
- **Homeowner Presence.** Whether the homeowner is *home* or *away*. We consider this factor to identify potential people who are likely to use or trigger the applet. This and the next factor can help identify if the applet is triggered by home members, visitors or others (outsider, *etc.*).
- **Visitor Presence.** This factor shows if the home has a visitor or not. It helps to identify potential people who are nearby and are able to use the applet or observe when the applet is activated, *i.e.,* observe the action. We considered this and the previous factor to understand if the applet is likely to be visible or used by the owner, visitors or others (outsider, *etc.*).

Usage Scenario: For each applet, we created deployment and usage contexts by considering all applicable combinations of the six contextual factors. For example, in an applet that connects a voice assistant like Alexa to a security camera: the trigger device (Alexa) can be located in the *living room, kitchen, bedroom, or bathroom*; the action device (camera) can be potentially located in the *living room, kitchen, or bedroom*[4]; the homeowner can be *absent, present and awake, or present and asleep*; a *visitor* may or may not be at home; and the applet can be run in the *morning, afternoon, or night*.

Violation Labeling: Violation types (e.g., integrity and secrecy) may vary in relevance across deployment and usage contexts. Generated scenarios can be grouped into two potentially overlapping sets for secrecy and integrity violations, each labeled as *violating* or *non-violating*. *Violating* means running the applet in the given context may potentially lead to an unintended access (integrity violation) or an unintended information flow (secrecy violation).

Dataset: For the 39 selected applets, we came up with 4104 applet deployment scenarios by considering all applicable combinations of the six contextual factors. All the generated scenarios were relevant for secrecy violations. But, only 3666 among the 4104 were relevant for integrity violations. This dataset was labeled by two of the authors and is used to (i) empirically investigate if the lattice-based approach is sufficient to accurately determine security and privacy violations in trigger-action smart-home applets (Section 4), and (ii) for evaluating our proposed learning framework (Section 5)

4 LATTICE-BASED MODEL LIMITATIONS

The lattice-based approach (Surbatovich et al., 2017) which uses a simple rule-based model (described in Section 2) to assign lattice labels and detect information flow violations in trigger-action applets, provides an efficient and scalable way of assessing the security and privacy issues in a large corpora of applets. However, it is prone to inaccurate detection (both false positives and false negative).

Cobb *et al.* (Cobb et al., 2020) enhanced this lattice-based model by adding two additional lattice labels for trusted people in the restricted physical space (*e.g.,* family members) and restricted online space (*e.g.,* household members who can control devices via an app). They found that about 57% of applets to be potentially violating compared to about 35% with the original lattice. However, when they manually inspected the violating applets, not all of the applets raised concerns leading them to conclude that:

[4]We consider locations for apps that are likely to occur. A camera in the bathroom is unlikely for most situations and users.

(a) semantic labels of triggers and actions can help find risky applets more accurately, and (b) contextual information (*e.g.,* device location) matters when evaluating such violations and impacts the performance of violation detection. Saeidi *et al.* (Saeidi et al., 2020) found that in addition to usage context, users' preferences also impact end-users perception of privacy and security violations in trigger-action applets.

Here, we replicate the analysis done by Cobb *et al.* by considering semantic labels of triggers and actions and contextual information to investigate the role of context in capturing the unintended accesses or information flows. We also show that in some cases contextual information alone may not suffice to accurately capture risks.

In particular, two authors independently reviewed each usage scenario, applying violation labels based on the context across multiple labeling batches. To ensure the validity of labeling for violation determination, in every step, the two authors discussed their assigned labels of each applet scenario in the batch and agreed on the final labels if their labels were not matched. As the authors progressed through multiple batches, the inter-rater reliability among them has consistently improved and we could achieve a high (Cohen's K ˜.8) in the final batch. Labeling was done under the assumption that applets are installed in a single resident one-bedroom apartment. Under this assumption, the resident will be the only trusted person for online and physical access.

Here, we compare the violation labels obtained through our context-based evaluation with those derived by applying the same rules used by Cobb *et al.* (Cobb et al., 2020) in the lattice-based analysis of the applets. Table 1 shows a comparison of the resulting secrecy and integrity violations. Overall, for secrecy violations, we found 1248 (out of 4104 unique applet scenarios) as violating using the lattice model. However, only 304 among those 1248 were labeled as secrecy violation when examining the scenario using contextual information. Similarly for integrity violations, we found 853 (out of 3666 unique applet scenarios) cases as violating using the lattice-based approach. However, when examined using contextual information, only 498 of them have been considered to be an integrity violation, with the remaining 355 being labeled as non-violation.

Table 1: Comparison of context-based and lattice-based detection of integrity and secrecy violations in the synthesized dataset. TP: True Positives, FP: False Positives

Secrecy			
Lattice	Context	Count (T=4104)	Per%
Violating (1248)	Violating (TP)	304	24%
	Non-violating (FP)	944	76% (23%)
Non-violating (2856)	Violating	99	3% (2%)
	Non-violating	2757	97%
Integrity			
Lattice	Context	Count (T=3666)	Per%
Violating (853)	Violating (TP)	498	58%
	Non-violating (FP)	355	42% (10%)
Non-violating (2813)	Violating	1212	43% (33%)
	Non-violating	1601	57%

The lattice-based approach over detected violations (*i.e.,* false positives), flagging 23% and 10% of all applet scenarios (See the last column in the Table 1) as secrecy and integrity violations respectively when compared context-based evaluation. This is somewhat unsurprising as the lattice-based approach can be expected to be more conservative. However, a significant number of cases that are considered violations when using context-based analysis were missed (*i.e.,* flagged as non-violations and hence false negatives) by the lattice based approach. Specifically, 99 secrecy cases (2% of all applet scenarios) and 1212 integrity cases (33% of all applet scenarios) were missed (see the last column in the Table 1). While some of the missed violations can be rectified by using finer grained lattice labels, our review found that others are a symptom of more fundamental limitations of the lattice-based approach. For instance, in a lattice-based approach, the lattice labels for the trigger and action are typically assigned independently. Such independent labeling, irrespective of the context of use, can lead to inaccurate violation detection.

Consider two groups of applets: *(i)* applets connecting a voice assistant (*e.g.,* Alexa) to online services (*e.g.,* Google Calendar) to enable end users to write into online services via a voice command (*e.g.,* adding a to-do list to Google Calendar), *(ii)* applets connecting a smart IoT device within the home (*e.g.,* smart lock) to online services (*e.g.,* Google Calendar) to notify or log the status of the IoT device (*e.g.,* when the smart lock gets unlocked, add an event to Google Calendar).

In both groups of applets, suppose given the contextual information, the integrity label for the action is *trusted* implying that only a trusted user (*e.g.,* the owner) can modify the Google Calendar. In this situation, if anyone but the homeowner triggers the app (interacts with the assistant or unlocks the lock), the lattice-based analysis will mark both groups of applets as violating applets (information flows from un-

trusted to trusted). While this kind of reasoning can capture the violations related to the first group of applets, it does not accurately model the second group of applets. For instance, if homeowners install (smart lock app) for surveillance purposes, they would like to monitor whenever the lock gets unlocked regardless of who is triggering the event. Hence, even if an unauthorized person (untrusted source) triggers writing to Google Calendar (a trusted resource), from a user perspective this is a valid usage and not a violation.

As these examples show, even if one were to integrate contextual information into the lattice-based model approach of (Surbatovich et al., 2017; Cobb et al., 2020), such a model still cannot accurately detect violations. In other words, contextual information alone may not lead to accurate decisions about violations; user perspectives must also be taken into account, and both trigger and action events need to be considered together for a more accurate evaluation.

To accommodate such multiple attribute evaluation, we propose to employ rule expression that enables us to evaluate many different attributes together. Further, to provide such personalization we investigate how learning-based techniques perform in identifying violations.

5 LEARNING DETECTION RULES

We explore how machine learning classification techniques can effectively learn end-user preferences to identify applet usage scenarios that may lead to violations based on contextual information. Recall from Section 4 how determining violations not only depends on the context of use, but also individuals' preferences. We propose a framework that can learn violation detection rules by taking into account both these factors to create a more personalized and flexible way to determine integrity and secrecy violations. More specifically, we examine the effectiveness of our learning framework to learn the violations from end-users' decisions on a set of applets and identify violations for unseen apps. Two criteria, inspired by prior work (Beckerle and Martucci, 2013; de Fortuny and Martens, 2015) in rule mining and extraction in access control, are used to assess our framework.

Classification Performance. The model's performance reflects how accurately the generated rules by the framework can classify usage scenarios as violating or non-violating.

Complexity. As our goal is to help end users by automatically learning violation detection rules, it is important that these rules not be overly complex to understand or to maintain. Prior literature (Smetters and Good, 2009; Bauer et al., 2009) showed that end users are more comfortable with fewer number of rules and those that are simpler to understand. We use rule complexity as a way to evaluate how effective our framework is in reducing the complexity of final rules in terms of their length and number.

5.1 Learning to Identify Violations

We define the problem of violation identification as a supervised classification problem. Specifically, we aim to learn a "rule" that maps a tuple of a trigger, action, and a set of contextual factors to binary labels specifying whether or not the use case is a violation.

Classifier Selection. Several popular supervised classification models such as Decision Tree, Random Forest, Support Vector Machines, and Logistic Regression can potentially be used to create a context-based prediction model (Sarker et al., 2019). Among these models, we decided to explore tree-based models such as Decision Tree and Random Forest, as they are easier to interpret as a set of rules and thus more suitable for our application. However, we also used a subset of other popular classification models, including Support Vector Machines and Logistic Regression, to baseline the performance of tree-based models and measure how well they perform compared to other models. Hence, for each classification problem (integrity and secrecy violation respectively), we investigated the above four algorithms.

Dataset. We used the synthesized dataset described in Section 3 to train and evaluate our machine learning models. The dataset consists of 39 applets and is divided into two subsets, each corresponding to a type of violation: integrity and secrecy. Each record (row) in the dataset includes 8 contextual factors (See Section 3).

Since the "Online Service Trustworthiness/Secrecy" feature applies only to applets connecting online services with IoT devices, we combined it with trigger/action location for online service events. This approach eliminated null values in the dataset for applets without online services. Table 2 shows the feature values in the datasets.

The final dataset is a set of tuples in the form of $< TS, AS, TL, AL, T, VP, HP, VL >$ where

- TS $\rightarrow$ trigger device/service,

Table 2: Features and their values in the final datasets.

Features
Trigger Device/Service (TS)[*] = {V: Voice command, WT: Weather or time, N: News-ish, DS: Sensing IoT device state, DS:S: Home security, DS:A: Appliances, DS:L: Lights, E: Environment sensing, IC: Intentional communication, P: Change personal device state, OAcc: Sensing online account state}
Action Device/Service (AS)[*] = {DS: Changing IoT device state, DS:S: Home security, DS:L: Lights, DS:A: Appliances, DS:P: Printer, E: Environment sensing, L: Log or notify, P: Change personal device state, OC: Outgoing communication}
Trigger Location (TL) = {Trusted/Private_Online_Service, Untrusted/Public_Online_Service, Kitchen, Living Room, Bathroom, Bedroom}
Action Location (AL) = {Trusted/Private_Online_Service, Untrusted/Public_Online_Service, Kitchen, Living Room, Bathroom, Bedroom}
Time (T) = {Morning, Afternoon, Night}
Visitor Presence (VP) = {Visitor, No-visitor}
Homeowner Presence (HP) = {Present-awake, Present-asleep, Absent}
Violating class label (VL) = {Violating, Non-violating}

[*] DS categories (S: Security, A: Appliances, L: Lights) are subtypes of IoT device states.

Table 3: Dataset. V: Violating, NV: Non-Violating.

Violation	V	NV	Total
Secrecy	403	3701	4104
Integrity	1710	1956	3666

- AS $\rightarrow$ action device/service,
- TL $\rightarrow$ trigger location,
- AL $\rightarrow$ action location,
- T $\rightarrow$ the time of the day,
- VP $\rightarrow$ visitor presence,
- HP $\rightarrow$ homeowner presence, and
- VL $\rightarrow$ violating class label

Table 3 shows the details of the dataset. For integrity violation, there are 1710 and 1956 records for violating (V) and non-violating (NV) classes, respectively. For secrecy violation, the dataset includes 403 violating (V) and 3701 non-violating (NV) instances. Finally, to prepare the datasets for the experiments, we transformed all categorical features into numeric values using Label Encoder method. For example, the values of the feature Time (Morning, Afternoon, and Night) gets translated into numeric values $(0,1,2)$. While one-hot encoding is a recommended encoding method for categorical variables, this method will impact the complexity of tree-based machine learning models by adding dummy variables with values of 0 and 1 and the resulting growth of trees in the direction of zero values. To be sure, besides numeric encoding, we used One-Hot Encoding and repeated all experiments. While we could achieve better performance with other machine learning models like SVM with one-Hot Encoding, we did not observe a significant difference in the performance of the RF model. However, one-hot encoding influenced the complexity of the learned rules and caused a significant decrease in the performance after application of rule reduction steps due to the presence of rules including zero values for dummy variables. Our findings are consistent with the experience of others (Gorman, 2017) that indicated one-hot encoding would not have better performance compared with a numeric encoding when the cardinality of categorical variables is smaller than 1000 for tree-based models.

Training Setup. The synthesized datasets for secrecy violations is unbalanced between the number of *violating* and *non-violating* cases. To address this, we applied random oversampling to the minority class to create a better-balanced dataset and improve classification performance (Chawla et al., 2002). Further, we used a nested cross-validation approach for model evaluation and hyper-parameter optimization that helps to overcome the problem of overfitting the training dataset. Specifically, the outer cross-validation procedure helps to evaluate the model on unseen data, while the inner cross-validation helps with finding optimal model parameters.

Within the outer cross-validation procedure, we split the dataset into $K = 5$ folds. In each iteration, we used the $K-1$ folds (80% of the entire dataset) for training the model and one fold (20% of the dataset) for testing it. The held-out fold helps to validate model performance on unseen data. We split the dataset using the Stratified-Group-K-Fold split technique to keep the distribution of both violating and non-violating samples the same in the training and testing splits. Specifically, we selected a subset of the applets (and all combinations of the contextual factors) for the training and testing split, such that each split had the same rate of violating and non-violating samples. We used the applet number that identifies a unique applet as the group factor. We computed the average performance scores across all K trials to report the final evaluation scores.

Next, in the inner cross-validation, we evaluated different hyper-parameter combinations to find the

Table 4: Classification results using different classification models.

	Secrecy			Integrity		
Model	Precision	Recall	F1	Precision	Recall	F1
RF	**0.93**	**0.91**	**0.91**	**0.77**	**0.75**	**0.75**
DT	0.93	0.92	0.92	0.75	0.76	0.75
SVM	0.90	0.89	0.90	0.66	0.65	0.65
LR	0.83	0.83	0.83	0.53	0.53	0.52

Table 5: Classification and complexity results using Random Forest (R#: The number of rules, V: Violating, NV: Non-Violating)

	Precision	**Recall**	**F1**	**WSC (V, NV)**	**R# (V, NV)**
Secrecy	0.93	0.91	0.91	2065.8, 2522.6	167, 197
Integrity	0.77	0.75	0.75	8340.2, 8382.8	629, 611

best-performing model. We used grid search to train each model on the training set and identify the optimal settings for each classifier model. To achieve that, we split the training dataset (*i.e.,* 80% of total data obtained from the outer cross-validation procedure) into K' subsets. Then we fit the model K' times on the $K'-1$ folds and evaluated it on the $K'th$ fold. Using Scikit-Learn GridSearchCV, we evaluated all hyper-parameter combinations and ran the K-fold-CV ($K'=5$) process for each of the combinations.

Performance Evaluation. We implemented our framework using Python and ran the experiments on a 64-bit macOS 10.12 machine having 16 GB RAM and Intel Core i5 processor. We conducted all experiments on the synthesized data set, which was labeled by two of the authors. We performed each experiment 10 times and report the average of the evaluation metrics in our experiments. We used *precision*, *recall*, and *F1-measure* metrics to measure the performance of the models' prediction.

Table 4 shows the details of the results for the best performance of each model (when testing on applets in the 20% data split). When predicting integrity violations, all models except Logistic Regression (LR) performed well, with Random Forest (RF) and Decision tree (DT) doing the best–an F1-measure of 0.75, followed by Support Vector Machine (SVM, F1-measure = 0.65). For secrecy violations, DT performed the best (F1-measure = 0.92), followed by RF (F1-measure = 0.91), and then SVM and LR with F1-measures of 0.90 and 0.83, respectively. These models had similar precision and recall values, showing that the models are well balanced in the number of false positives and false negatives they generate. Given the "good" performance of the classification models, we believe that a learning framework is a viable solution in learning from users' security and privacy preferences in a given use context from a set of applets and applying this learning to identify violations in other applets.

For both types of violations the tree-based learning models performed better indicating that violation detection based on the usage context is not a linear problem. The RF model consistently demonstrated higher precision in detecting integrity violations, while both models (RF and DT) performed similarly for secrecy violations. Although secrecy violation dataset is imbalanced, RF showed slightly better generalization than DT due to its ensemble structure, which reduces variance and helps prevent overfitting. However, it is important to note that RF does not inherently address class imbalance and additional techniques might be necessary to improve fairness. We focus on the RF model in the rest of the paper due to its overall stability and performance.

5.2 Violation Rule Complexity

Machine learning systems typically appear as "black boxes" to end users (Kulesza et al., 2015) leaving users unable to understand their behavior, which creates a knowledge gap regarding model outputs (Miller, 2019). In this paper we chose to investigate tree-based learning models, which make it easier to interpret the model's "decision-making" as conditional (IF-THEN) rules. Another advantage is that end users may also extract and employ these rules in existing frameworks for controlling IFTTT applets (*e.g.,* SOTERIA (Celik et al., 2018), IoTGuard (Celik et al., 2019), Expat (Yahyazadeh et al., 2019)) that require users to provide a set of rules.

Several previous works have used Decision Tree (DT) and Random Forest (RF) algorithms for anomaly detection or access control policy mining (Bui and Stoller, 2020) for similar reasons. Here, we selected RF as it was the model with the best performance in our experiments (See Table 4).

However, RF models can create very large rule sets. Therefore, to create a manageable set of violation detection rules for human consumption, we

reduce the complexity of extracted rules from the trained RF model through a series of steps. We use Weighted Structural Complexity (WSC) as a metric to measure the complexity of learned rules. WSC was introduced for studying complexity of mined polices in Role-Based Access Control (RBAC) (Molloy et al., 2010) and Attribute-Based Access Control (ABAC) (Xu and Stoller, 2014) systems. Informally, for a given rule set, WSC is a sum of the WSC for all of its rules. For each rule, WSC is a weighted sum of the number of elements (features) in that rule. Here, we set the weight for all features to be 1.

As Table 5 shows, the best performing RF models contain about 364 and 1240 rules (totaling rules for violating and non-violating cases) for secrecy and integrity detection, respectively. The WSC metric for these rules is around 4588 and 16723, respectively. Given the high complexity of the final rules, we aim to reduce both the number of rules and the WSC score through a series of reduction steps applied to the extracted rules for each prediction class.

Extracting Rules. We convert each decision tree in the RF model into a set of rules by traversing each distinct path from the root to a leaf node representing a prediction class (*violating, non-violating*). Each rule will be in the form of a conditional statement (Equation 1), where each condition is associated with a node on the path. We split the final ruleset into two sets, where each set includes the rules for each prediction class. One example of an extracted rule from the *violating* class for integrity violation is shown in Equation 2.

$$R = \left[\begin{array}{l} IF\ (f_1 < v_1)\ AND\ (f_2 < v_2)\ AND\ \ldots \\ \qquad THEN\ C \\ where\ f_i\ is\ a\ contextual\ factor, \\ v_i \in (0, v_{max_i})\ and\ v_{max_i}\ is\ the \\ maximum\ encoded\ numeric\ value\ of\ f_i, \\ and\ C\ is\ the\ prediction\ class \end{array}\right] \quad (1)$$

Transforming Rules. In this step, to make the application and evaluation of the reduction steps easier, we transform each rule into a form that enables us to compare their attributes. In this step, we replace each condition in the rule of the form ($f_i < or >= v_i$), where v_i is a splitting point, with a triple of $F_i = [f_i, l_i, u_i]$, where l_i and u_i are the lower-bound and upper-bound values meaning the minimum and maximum numeric values that the feature f_i can have, respectively. The final rule will be a vector $< F_1, F_2, ..., F_n, C >$ where C is the prediction class. The transformation of the rule in Equation 2 is shown in Equation 3. For instance, the condition $(Homeowner - Presence <= 0.5)$ in $R1$ indicates that this factor only takes 0 value meaning 'the homeowner is present and awake.' This condition will be transformed to a triple in the form of $(Homeowner - Presence, 0, 0)$.

Reducing Redundant Features. The Random Forest approach generates each decision tree based on randomly selected features or combinations of features at each node in each tree (Breiman, 2001). That is, some features can be selected multiple times to split a node. Hence, each rule can have multiple similar features. To reduce the complexity of the rules, in this step we merge the redundant features by selecting the least upper bound and the greatest lower bound of the feature values. For instance, in rule R1 in Equation 3, Trigger-Semantic is a redundant feature that can be replaced with a single triple *(Trigger-Semantic, 4, 8).*

$$\begin{aligned} R1 = [\ & \text{IF (Homeowner - Presence} \leq 0.5) \\ & AND\ (Trigger\text{ - }Semantic > 3.5) \\ & AND\ (Trigger\text{ - }Semantic \leq 8.5) \\ & AND\ (Visitor\text{ - }Presence \leq 0.5) \\ & AND\ (Trigger\text{ - }Location \leq 4.0) \\ & THEN\ prediction\text{ - }class = 1\ ] \end{aligned} \quad (2)$$

$$\begin{aligned} R1 = [\ & (Homeowner\text{ - }Presence, 0, 0), \\ & (Trigger\text{ - }Semantic, 4, 8), \\ & (Trigger\text{ - }Semantic, 0, 8), \\ & (Visitor\text{ - }Presence, 0, 0), \\ & (Trigger\text{ - }Location, 0, 4), \\ & prediction\text{ - }class = 1\ ] \end{aligned} \quad (3)$$

Reducing Subset Rules. The RF approach creates multiple trees, and it is possible to have similar rules across multiple trees or for a rule extracted from one tree to be a subset of a rule extracted from a different tree. Given two rules R_1 and R_2, R_1 is a subset of rule R_2, if the interval of values of each feature in R_1 is a subset of interval of values of the same feature in R_2. In such a case, having both rules R_1 and R_2 is redundant. In this step, we eliminate such redundant rules by removing the less restrictive one, in this case R_1.

Reducing Similar Rules: In this step we search for similar rules in the ruleset and prune rules such that it does not decrease the quality of the ruleset. We define a ruleset quality metric by combining two metrics, F1-measure and WSC, based on Van Rijsbergen's effectiveness measure (Van Rijsbergen, 1979), as was also done by (Karimi et al., 2021). The ruleset quality metric is defined as follows:

Table 6: Comparison of performance and complexity results of the lattice-based approach and violating and non-violating rules before and after reduction. (R#: The number of rules, V: Violating, NV: Non-Violating)

	Lattice-based Approach				
	Precision	Recall	F1	WSC (V, NV)	R# (V, NV)
Secrecy	0.64	0.64	0.54	due to simplicity of the lattice	
Integrity	0.54	0.58	0.47	the complexity is too small	
	Original Rules				
	Precision	Recall	F1	WSC (V, NV)	R# (V, NV)
Secrecy	0.93	0.91	0.91	2065.8, 2522.6	167, 197
Integrity	0.77	0.75	0.75	8340.2, 8382.8	629, 611
	Post-reduction Rules				
	Precision	Recall	F1	WSC (V, NV)	R# (V, NV)
Secrecy	0.81	0.81	0.80	432.2, 635.4	67, 76
Integrity	0.74	0.73	0.73	2194.2, 2220	251,242

$$Q = \left(\frac{\alpha}{F1-measure} + \frac{1-\alpha}{\Delta WSC}\right)^{-1} \quad (4)$$

In the above equation, α shows the importance of F1-measure metric over the ruleset complexity, and ΔWSC shows the relative reduction in WSC score of the ruleset compared to the original ruleset. In each step, we select a rule from one tree and find a similar rule in other trees.

We measure the similarity between two rules $R1$ and $R2$ based on Jaccard similarity which is defined as follows: $J(R1,R2) = |R1 \cap R2|/|R1 \cup R2|$. We define two rules as similar if their Jaccard similarity is more than 80%, which means the size of their common feature values is more than 80% of the size of the union of their feature values[5]. To prune similar rules in each step, we pick two rules from two different trees in the RF model and calculate their similarity. If the similarity score is more than 0.8, we decide to remove one of the two rules depending on how much each of the rules will improve the quality score of the rules set.

Combining Rules: While in the *Reducing Subset Rules* step, we aimed to remove the rules that are a subset of other rules in the ruleset, it is likely to have rules with similar value intervals for all features except one. This happens because, for each rule, the algorithm only searches the subset rule until it finds the first one. Hence, the algorithm will not find all subset rules for each rule. In the last step, we calculate the union of each two rules in the set having the same set of features and are different in interval value of only one feature. We add the union rule to the ruleset and remove the other two rules.

Rule Reduction Evaluation. We assess rule reduction by comparing the final reduced rules with the original rules (extracted rules from the RF model before rule reduction) based on rule complexity (number of rules and WSC), similarity, and prediction performance.

Table 6 represents the precision, recall, F1-measure, WSC score, and the number of rules before and after reduction steps for both secrecy and integrity violations. Our proposed rule reduction process could reduce the WSC and the number of rules by about 77% (from 4588 to 1067) and 61% (from 364 to 143), respectively, while still obtaining 88% of the original F1-measure for secrecy violations. For integrity violations we could decrease the WSC by about 74% (from 16723 to 4414), and the number of rules by about 60% (from 1240 to 492) on average, and achieve about 97% of the original F1-measure. Comparing the final rules before and after rule reduction steps with the lattice-based approach, while the lattice-based approach employs simple rules (recall from Section 2) to identify violations, it does not perform well when predicting violations obtaining F1-measure 0.54 and 0.47 for predicting secrecy and integrity violations, respectively.

These findings suggest that we can simplify learned rulesets with minimal performance loss. Future work will explore how users can fine-tune these rules to their preferences and assess their maintainability as new applets are installed.

6 DISCUSSION

6.1 Deployment Challenges

Collecting Contextual Information. Our proposed approach requires usage context data (*e.g.,* location, time, presence *etc.*). The interplay among users, the physical environment, and devices creates a dynamic context, making it challenging to identify and collect relevant information. While smart home devices can likely provide this data, collecting it in

[5]Other work in access control mining (Karimi et al., 2021) have used a 50% threshold, but we chose to be more conservative to preserve prediction performance.

a real system may not always be easy or possible. Existing frameworks like SmartThings enable users to define and collect some contextual information allowing them to set variables such as "home mode" or designate device locations on a map. Similarly, the presence of visitors can be detected using motion sensors or features in smart-home apps (*e.g.,* party mode) (Chi et al., 2020). However, such functionality varies by IoT frameworks and may not be available in all frameworks. Additionally, users may opt not to provide or update this information. So, along with our rule learning module, we need to include options for end users to identify such common situations depending on the existing framework or include a dedicated module to collect such information from sensors.

Collecting Labelled Data for Training. Another deployment challenge is access to enough training data labelled by the the end user. Such data can potentially be collected from the end-user during the first few days or weeks of training phase post deployment. For instance, in a similar work HAWatcher (Fu et al., 2021), the required data (22,655 events) is collected during the three weeks of training phase. In another work AEGIS (Sikder et al., 2019), the authors created their required dataset consisting of over 55000 events in a 10-day period for anomaly detection in smart homes. Data collection can include collecting contextual information and user's preferences of allowing or not allowing an applet to run in that context.

Enforcement Mode. Violation detection approaches can be categorized into inline or offline based their enforcement mode. Inline approaches prevent potential violations by intercepting requested actions at runtime necessitating app or framework instrumentation to be able to intercept actions in real-time. But, this mode also provides easy access to current contextual information. This approach taken by previous works such as Expat (Yahyazadeh et al., 2019), and IoTGuard (Celik et al., 2019).

On the other hand, our approach can be deployed in a monitoring or advisory mode, collecting contextual data through sensors and app logs, and notifying users of potential violations. This reduces the burden of having to instrument the apps or the frameworks but does not prevent or block actions that can cause violations. AEGIS (Sikder et al., 2019) is an example of prior work using this deployment mode.

Multiple User Home Environment. We evaluated our approach with a synthesized dataset for a single resident home. However, smart homes configured for multiple residents face greater challenges. For instance, different roles in the home (*e.g.,* kids, teenagers, and spouses) would have different levels of access to devices. Prior work, Kratos (Sikder et al., 2020), has proposed an access control for such an environment. This framework introduces a formal policy language enabling users to define access control policies with their priorities. However, this approach still needs manual effort by the end user to define the policies. Further, gathering sufficient contextual information to identify which user is interacting with the applet complicates the deployment of the learning framework. One solution could be using multiple sensors to collect contextual data. Previous research on daily activity recognition (Dahmen et al., 2017) has shown it's possible to learn the location and activities of home members using data from smart sensors. However, both the learning approach and the contextual factors used may need to change significantly to accommodate multi-user home environments.

6.2 Future Work

Learned Rules as Predefined Rules. Several studies have proposed solutions for detecting violations in smart-home environments (Celik et al., 2019; Yahyazadeh et al., 2019). These solutions need manually defined violation detection rules or policies. Some of these solutions enable users to define such policies using policy languages (Yahyazadeh et al., 2019). However, defining such policies is a complex task for users. For instance, study (Cao and Iverson, 2006) shows that defining intended access as an access control rule is a complex task with a heavy cognitive workload for users. Another study (Mazurek et al., 2010) shows that users find specifying fine-grained access control policies a difficult task. In these cases, learned violation detection rules can be used as predefined or suggested rules in those frameworks. This requires studying the feasibility of modifying rules learned for one user for use by another user and efficient mechanisms for doing so.

Dynamic Security Labels. Discussion in Section 4 showed that static (context agnostic) lattice labels used by prior lattice-based approaches (Surbatovich et al., 2017; Cobb et al., 2020) could not always determine the violations associated with trigger-action apps accurately. The context of use varies among different individuals, which would determine the appropriate lattice label for an individual and their specific use scenario. For example, a trigger device in a living room might have a trusted label when

only family members and an untrusted label when having a guest. In another example, the owner may assign a trusted label even if a guest is at home because she/he trusts the guest. One approach could be learning the lattice labels based on the usage context and individual preferences. In this approach, the labels of the triggers and actions would change based on the detected context and customized based on individuals' preferences.

Actual Violations vs Mis-Classifications. The proposed framework based on usage context has shown promising results in detecting violations. However, further research is needed to determine if users perceive misclassified violations as actual violations. A large-scale user study can help identify these preferences and determine if the misclassified violations are actual violations for most users. Even if the study were to show that the misclassified violations are not actual violations, it emphasizes the importance of users' preferences in determining violations. Therefore, it's crucial to allow users to modify rules or train the model based on their needs, creating an adaptive learning environment for better performance and outcomes. This approach will enable us to create a learning environment that adapts to users' preferences, ultimately leading to better performance and outcomes.

7 RELATED WORK

Our work aims to help end users detect undesirable or unintended access to information and applications/devices enabled through the use of trigger-action frameworks in smart homes.

Much work has been done to identify security and privacy risks in smart homes and more generally in IoT eco-systems. Among those, one of the most closely related to our work is by Surbatovich *et al.* (Surbatovich et al., 2017), who employed lattice-based information flow analysis to evaluate the potential integrity and secrecy violations in a 20K IFTTT dataset. They employed security labels for triggers and actions and determined the violations using information analysis. Their result showed more than 50% of the IFTTT applets would potentially cause a violation. More recently Cobb *et al.* (Cobb et al., 2020) adjusted the security lattice by adding two more fine-grained labels to the lattice. While they showed that IFTTT applets are not as risky as initially portrayed in (Surbatovich et al., 2017), they found that the lattice-based approach still has significant false positives and negatives. They acknowledged that more contextual information is needed to better evaluate the secrecy and integrity violations in smart homes. Our work shows that even with contextual information, determining violations depends on users' preferences and isn't straightforward.

Other works have addressed security and privacy risks in smart-home environments using model checking. For instance, SOTERIA (Celik et al., 2018) uses the state model of an individual or set of apps to check safety, security, and functional properties and identify violations. IoTGuard (Celik et al., 2019) proposes a dynamic policy-based enforcement system to evaluate user-defined safety, security, and functional properties using model checking. SAFECHAIN (Hsu et al., 2019) uses model-checking to discover the privilege escalation and privacy leakage threats across IoT apps. While SAFECHAIN uses the security labels proposed by Surbatovich to define the threats, it can consider the actual attribute values of the rules and decrease the false positive rate in privacy leakage detection compared to the lattice-based approach. However, all the above solutions rely on static predefined rules. If these rules are predefined by system designers, then end user preferences will be unaccounted for. On the other hand, tasking end users to define such rules would be burdensome. Our work would ease making policy rules for these existing solutions.

There are also works that focused on implicit and explicit interaction across IoT applications. For instance, IoTMon (Ding and Hu, 2018) uses risk analysis to discover the potential physical channel interaction across IoT apps. In another work, Wang *et al.* (Wang et al., 2019) propose a tool, iRuler, to identify vulnerabilities among trigger-action rules. They use the NLP technique to find the information flows between actions and triggers across IFTTT applets. HomeGuard (Chi et al., 2020) leverages SMT solvers to discover cross-app interference threats. This solution ranks identified threats using risk analysis, allowing end users to evaluate them by risk scores. However, it does not detect threats in individual applets. HAWatcher (Fu et al., 2021) uses data mining to predict anomalies in smart IoT device behavior but only detects malfunctions and assumes all installed smart home apps are benign.

Further, most of the above solutions did not pay attention to the ease of use of their systems. Our goal was to reduce the complexity of the rules and generate more explainable rules for users to deploy and manage. The rule reduction techniques that we used are similar to those used in other research areas like access control systems (*e.g.,* (Karimi et al., 2021)) to extract simple access control policies.

8 CONCLUSION

This paper presents a learning framework for predicting integrity and secrecy violations in smart-home IFTTT applets. The framework considers contextual factors (location, time, app trustworthiness, and presence of visitors and homeowners) that impact these violations. Trained on a synthesized dataset, the proposed framework was able to detect integrity violations with an average of 0.77 precision, 0.75 recall, and 0.75 F1-measure, and secrecy violations with an average of 0.93 precision, 0.91 recall, and 0.91 F1-measure. We also showed that a learning framework that accounts for contextual factors and users' preferences performs better than using lattice-based approaches. Future work will explore how mined rules can help users refine their privacy and security preferences and how these rules can be transferred to other users with similar privacy attitudes for scalability.

Acknowledgements

Much of the work was completed while Mahsa Saeidi was a Ph.D. student at Oregon State University. We thank Souti Chattopadhyay, Matt Jansen, and Turguy Caglar for their efforts in labeling preliminary versions of the synthesized applet dataset.

REFERENCES

Bastys, I., Balliu, M., and Sabelfeld, A. (2018). If this then what?: Controlling flows in iot apps. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1102–1119. ACM.

Bauer, L., Cranor, L. F., Reeder, R. W., Reiter, M. K., and Vaniea, K. (2009). Real life challenges in access-control management. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 899–908.

Beckerle, M. and Martucci, L. A. (2013). Formal definitions for usable access control rule sets from goals to metrics. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, pages 1–11.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Bui, T. and Stoller, S. D. (2020). A decision tree learning approach for mining relationship-based access control policies. In *Proceedings of the 25th ACM Symposium on Access Control Models and Technologies*, pages 167–178.

Cao, X. and Iverson, L. (2006). Intentional access management: Making access control usable for end-users. In *Proceedings of the second symposium on Usable privacy and security*, pages 20–31.

Celik, Z. B., McDaniel, P., and Tan, G. (2018). Soteria: Automated iot safety and security analysis. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 147–158.

Celik, Z. B., Tan, G., and McDaniel, P. D. (2019). Iotguard: Dynamic enforcement of security and safety policy in commodity iot. In *NDSS*.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority oversampling technique. *Journal of artificial intelligence research*, 16:321–357.

Chi, H., Zeng, Q., Du, X., and Yu, J. (2020). Cross-app interference threats in smart homes: Categorization, detection and handling. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 411–423. IEEE.

Cobb, C., Bhagavatula, S., Garrett, K. A., Hoffman, A., Rao, V., and Bauer, L. (2021). " i would have to evaluate their objections": Privacy tensions between smart home device owners and incidental users. *Proc. Priv. Enhancing Technol.*, 2021(4):54–75.

Cobb, C., Surbatovich, M., Kawakami, A., Sharif, M., Bauer, L., Das, A., and Jia, L. (2020). How risky are real users'{IFTTT} applets? In *Sixteenth Symposium on Usable Privacy and Security ({SOUPS} 2020)*, pages 505–529.

Dahmen, J., Thomas, B. L., Cook, D. J., and Wang, X. (2017). Activity learning as a foundation for security monitoring in smart homes. *Sensors*, 17(4):737.

de Fortuny, E. J. and Martens, D. (2015). Active learning-based pedagogical rule extraction. *IEEE transactions on neural networks and learning systems*, 26(11):2664–2677.

Denning, D. E. (1976). A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243.

Ding, W. and Hu, H. (2018). On the safety of iot device physical interaction control. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 832–846.

Fernandes, E., Paupore, J., Rahmati, A., Simionato, D., Conti, M., and Prakash, A. (2016). Flowfence: Practical data protection for emerging iot application frameworks. In *25th {USENIX} security symposium ({USENIX} Security 16)*, pages 531–548.

Fu, C., Zeng, Q., and Du, X. (2021). {HAWatcher}:{Semantics-Aware} anomaly detection for appified smart homes. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 4223–4240.

Gorman, B. (2017). Visiting: Categorical features and encoding in decision trees. https://towardsdatascience.com/visiting-categorical-encoding-techniques-in-decision-trees-74f40d3a5d3d. Accessed: 2025-04-28.

He, W., Golla, M., Padhi, R., Ofek, J., Dürmuth, M., Fernandes, E., and Ur, B. (2018). Rethinking access control and authentication for the home internet of things

(iot). In *27th USENIX Security Symposium (USENIX Security 18)*, pages 255–272.

He, W., Zhao, V., Morkved, O., Siddiqui, S., Fernandes, E., Hester, J., and Ur, B. (2021). Sok: Context sensing for access control in the adversarial home iot. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 37–53. IEEE.

Hsu, K.-H., Chiang, Y.-H., and Hsiao, H.-C. (2019). Safechain: Securing trigger-action programming from attack chains. *IEEE Transactions on Information Forensics and Security*, 14(10):2607–2622.

IFTTT (2019). If this, then that (ifttt). https://www.ifttt.com/. Accessed: 2025-04-28.

Jin, H., Liu, G., Hwang, D., Kumar, S., Agarwal, Y., and Hong, J. I. (2022). Peekaboo: A hub-based approach to enable transparency in data processing within smart homes. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 303–320. IEEE.

Karimi, L., Aldairi, M., Joshi, J., and Abdelhakim, M. (2021). An automatic attribute based access control policy extraction from access logs. *IEEE Transactions on Dependable and Secure Computing*.

Kulesza, T., Burnett, M., Wong, W.-K., and Stumpf, S. (2015). Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th international conference on intelligent user interfaces*, pages 126–137.

Lee, H. and Kobsa, A. (2016). Understanding user privacy in internet of things environments. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 407–412. IEEE.

Lee, H. and Kobsa, A. (2017). Privacy preference modeling and prediction in a simulated campuswide iot environment. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 276–285. IEEE.

Mazurek, M. L., Arsenault, J., Bresee, J., Gupta, N., Ion, I., Johns, C., Lee, D., Liang, Y., Olsen, J., Salmon, B., et al. (2010). Access control for home data sharing: Attitudes, needs and practices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 645–654.

McLaughlin, S. (2021). Ifttt pro offers advanced features for a monthly subscription. https://techxplore.com/news/2020-09-ifttt-pro-advanced-features-monthly.html. Accessed: 2025-04-28.

Microsoft Corporation (2019). Microsoft flow: Automate processes and tasks. https://flow.microsoft.com/. Accessed: 2025-04-28.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38.

Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S., and Lobo, J. (2010). Mining roles with multiple objectives. *ACM Transactions on Information and System Security (TISSEC)*, 13(4):1–35.

Naeini, P. E., Bhagavatula, S., Habib, H., Degeling, M., Bauer, L., Cranor, L. F., and Sadeh, N. (2017). Privacy expectations and preferences in an iot world. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, pages 399–412.

Saeidi, M., Calvert, M., Au, A. W., Sarma, A., and Bobba, R. B. (2020). If this context then that concern: Exploring users' concerns with ifttt applets. *arXiv preprint arXiv:2012.12518*.

Sarker, I. H., Kayes, A., and Watters, P. (2019). Effectiveness analysis of machine learning classification models for predicting personalized context-aware smartphone usage. *Journal of Big Data*, 6(1):1–28.

Sikder, A. K., Babun, L., Aksu, H., and Uluagac, A. S. (2019). Aegis: A context-aware security framework for smart home systems. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 28–41.

Sikder, A. K., Babun, L., Celik, Z. B., Acar, A., Aksu, H., McDaniel, P., Kirda, E., and Uluagac, A. S. (2020). Kratos: Multi-user multi-device-aware access control system for the smart home. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 1–12.

Smetters, D. K. and Good, N. (2009). How users use access control. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, pages 1–12.

Surbatovich, M., Aljuraidan, J., Bauer, L., Das, A., and Jia, L. (2017). Some recipes can do more than spoil your appetite: Analyzing the security and privacy risks of ifttt recipes. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1501–1510. International World Wide Web Conferences Steering Committee.

Van Rijsbergen, C. J. (1979). Information retrieval. 2nd. newton, ma.

Wang, Q., Datta, P., Yang, W., Liu, S., Bates, A., and Gunter, C. A. (2019). Charting the attack surface of trigger-action iot platforms. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 1439–1453.

Washington State University (2019). Wsc datasets. http://casas.wsu.edu/datasets/. Accessed: 2025-04-28.

Xu, Z. and Stoller, S. D. (2014). Mining attribute-based access control policies. *IEEE Transactions on Dependable and Secure Computing*, 12(5):533–545.

Yahyazadeh, M., Podder, P., Hoque, E., and Chowdhury, O. (2019). Expat: Expectation-based policy analysis and enforcement for appified smart-home platforms. In *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies*, pages 61–72.

Zapier Inc. (2019). Zapier: Automate workflows. https://zapier.com/. Accessed: 2025-04-28.