TKPERM: Cross-platform Permission Knowledge Transfer to Detect Overprivileged Third-party Applications

Faysal Hossain Shezan
University of Virginia
fs5ve@virginia.edu

Kaiming Cheng
University of Virginia
kc4jd@virginia.edu

Zhen Zhang
Johns Hopkins University
zzhen1@jhu.edu

Yinzhi Cao
Johns Hopkins University
yinzhi.cao@jhu.edu

Yuan Tian
University of Virginia
yt2e@virginia.edu

***Abstract*—Permission-based access control enables users to manage and control their sensitive data for third-party applications. In an ideal scenario, third-party application includes enough details to illustrate the usage of such data, while the reality is that many descriptions of third-party applications are vague about their security or privacy activities. As a result, users are left with insufficient details when granting sensitive data to these applications. Prior works, such as WHYPER and AutoCog, have addressed the aforementioned problem via a so-called permission correlation system. Such a system correlates third-party applications' description with their requested permissions and determines an application as overprivileged, if a mismatch between the requested permission and the description is found. However, although prior works are successful on their own platforms, such as Android eco-system, they are not directly applicable to new platforms, such as Chrome extensions and IFTTT, without extensive data labeling and parameter tuning.**

In this paper, we design, implement, and evaluate a novel system, called TKPERM, which transfers knowledges of permission correlation systems across platforms. Our key idea is that these varied platforms with different use cases—like smartphones, IoTs, and desktop browsers—are all user-facing and thus allow the knowledges to be transferrable across platforms. Particularly, we adopt a greedy selection algorithm that picks the best source domains to transfer to the target permission on a new platform. TKPERM achieves 90.02% overall F1 score after transfer, which is 12.62% higher than the one of a model trained directly on the target domain without transfer. Particularly, TKPERM has 91.83% F1 score on IFTTT, 89.13% F1 score on Chrome-Extension, and 89.1% F1 score on SmartThings. TKPERM also successfully identified many real-world overprivileged applications, such as a gaming hub requesting location permissions without legitimate use.

I. INTRODUCTION

Permission-based access control is ubiquitously used in a variety of platforms—such as Android [4], Chrome Extension [3], IFTTT [5] and Samsung SmartThings [6]—to restrict the access of third-party applications to user private information. Take the Android platform for example. Access to the device location requires special permissions that are approved by users during the installation time of a third-party application. Similarly, on the Chrome platform, if a user installs an extension that requires access to the device location, the user needs to approve it before installation.

One important task of third-party applications on permission-based access control system is to provide enough knowledge for the user, essentially the decision-maker while granting these permissions, so that he or she can understand the rationale behind those applications requesting certain permissions. Such knowledge is usually in the format of a human-readable language that describes the app's functionality, and implies the connection to the permission. For example, an Android app, which describes itself to provide local weather information to users, justifies its request of the location permission as local weather needs the user's location.

Many prior works, such as WHYPER [37], AutoCog [40], and SmartAuth [49], have proposed to parse and understand such provided knowledge from applications, correlate it with requested permissions, and detect whether applications are requesting for unnecessary user privacy information. Such a *permission correlation system* is useful and successful to determine and detect overprivileged applications with unexpected permissions, such as a weather application requesting your contact information without any justifications. One can deploy permission correlation systems at either the client-side as an alert to the end user or the server-side that filters overprivileged applications with unexpected permissions.

Despite the success of existing permission correlation systems, one open question is that permission-based platforms are diverse, ranging from Android to Chrome extension and Internet of Things (IoT), while existing permission detection systems are all limited to one platform. For example, WHYPER [37] and AutoCog [40] focus on only Android, and SmartAuth [49] on only Samsung SmartThings. It would require much human work, such as data labeling and parameter tuning, to build a customized correlation system for each newly-emerged platform. More importantly, these newly-emerged platforms, usually having a relatively small number of third-party applications, may not have enough data for building a good correlations system with reasonable accuracy.

In this paper, inspired by prior successful use cases of transfer learning in image recognition systems [16], [34], [57], we

Network and Distributed Systems Security (NDSS) Symposium 2020
23-26 February 2020, San Diego, CA, USA
ISBN 1-891562-61-4
https://dx.doi.org/10.14722/ndss.2020.24287
www.ndss-symposium.org

explore the idea of transferring permission knowledge across platforms so that one can easily build a new correlation system for a different platform with prior knowledge. Our *key* insight is that while these platforms are varied with different use cases, like smartphones, IoTs, and desktop browsers, they are all user-facing, thus sharing certain aspects that are transferrable across platforms. Particularly, we propose a system, called TKPERM, to transfer semantics and permission correlation knowledge from one source platform to a target one, build the model using the transferred knowledge, and then integrate the model into a permission correlation system.

First, TKPERM transfers semantics knowledge, such as the semantic meanings of words, from one platform to another. The observation here is that many third-party applications on these platforms are of the same purpose or with similar descriptions. For example, both Android and Chrome extension have weather and proxy applications; similarly, birthday reminder applications exist in both IFTTT and Android platforms. Therefore, TKPERM can transfer this semantic knowledge, particularly in the form of word embeddings—that is, those words that share the same or similar meaning are close in the embedding space, and such similarity is transferred across platforms.

Second, TKPERM transfers permission correlation knowledge, such as the relationship of certain permissions with certain descriptions, from one platform to another. The observation here is that different platforms, though with usages, sometimes share certain permissions. For example, the location permission exists in both Android and Chrome extension; calendar access is required on both IFTTT and Android. Such permission knowledge, especially the correlation with application descriptions, is transferred by TKPERM in the form of parameters in neural network layers that are close to the input layer.

While the idea of transferring semantics and permission knowledge is intuitively simple, the challenge is that these permission-based platforms, though sharing some knowledge in common, are indeed different. Take permission knowledge for example. Android has two types of location permissions, one coarse-grained and the other fine-grained, but Chrome extension only has one location permission. Some permissions, such as evernote trigger on IFTTT, are platform-specific with no counterpart on other platforms.

TKPERM tackles the problem in platform differences via two steps. On the one hand, TKPERM fine-tunes the model transferred from the source platform using a small number of target domain data so that the subtle difference between platforms can be mitigated. Specifically, TKPERM freezes layers close to the input to preserve transferred knowledge and at the same time adjusts these layers close to output using such fine-tuning data to make the transferred model optimized for the target platform.

On the other hand, TKPERM introduces a greedy selection algorithm to choose the knowledge that is best suited for the target permission. Specifically, we define a domain as all the applications with a certain permission on a certain platform. TKPERM uses a greedy domain selection algorithm to choose the domains on the source platform with the highest F1 score and continue to add more domains on the source platform until the F1 score decreases. By doing so, TKPERM can select the best source domain combination for each given target and apply them in the transfer procedure.

We implemented a prototype of TKPERM that transfers knowledge from Android to three different platforms, which are IFTTT, Chrome extension, and SmartThings. Our source model, essentially a neural network, is built from 36,193 manually-labeled sentences of 1,234 Android apps and has 84.2% average F1 score on nine domains. The overall F1 score of the transferred models on three platforms is 90.02%, with 91.83% on IFTTT, 89.13% on Chrome Extension, and 89.1% on SmartThings.

Our target model built via the transfer learning performed by TKPERM successfully identifies 329 overprivileged applications in total. Examples of such overprivileged applications are like that a gaming hub Chrome extension requesting location permissions without legitimate use and a weather forecast recipe on IFTTT requesting access to a user's BMW car. We have responsibly reported these issues of overprivileges to these application developers if we can find their contact information somewhere online. We received some positive feedback, and we are still in the process of filing the issues and negotiating these issues with application developers.

We make the following contributions in designing and implementing TKPERM:

- We are the first to propose a generic framework, called TKPERM, which transfers knowledge between permission-based platforms. We implemented a prototype of TKPERM that transfers permission knowledge from Android to three different platforms.
- We manually labeled 36,193 sentences on Android, 4,705 on Chrome-extension, 666 on IFTTT, and 292 on SmartThings. We have publicly released our labeled dataset on https://drive.google.com/drive/folders/1Yfnz-ZpBpL8lftYIdM6JtH-QKE88NcSX?usp=sharing to facilitate open science.
- Our evaluation of TKPERM shows that TKPERM can successfully transfer knowledge from the Android platform to IFTTT, Chrome-Extension, and SmartThings with high F1 score. The transferred models find 329 overprivileged applications across these three platforms.

The rest of the paper is organized as follows. Section II introduces a motivating example of using transfer learning in our problem setting, Section III describes how transfer learning works, Section IV demonstrates the four components of TKPERM, Section V illustrates implementation of TKPERM, Section VI evaluates our system's performance, we include a case study of overprivileged applications from the three different platforms in Section VII, we discuss some limitations of our work in Section VIII, we compare TKPERM with the existing research works in Section IX, finally we conclude our paper in Section X.

II. A MOTIVATING EXAMPLE

In this section, we illustrate our transfer learning idea with a real-world Google Chrome extension. The Chrome

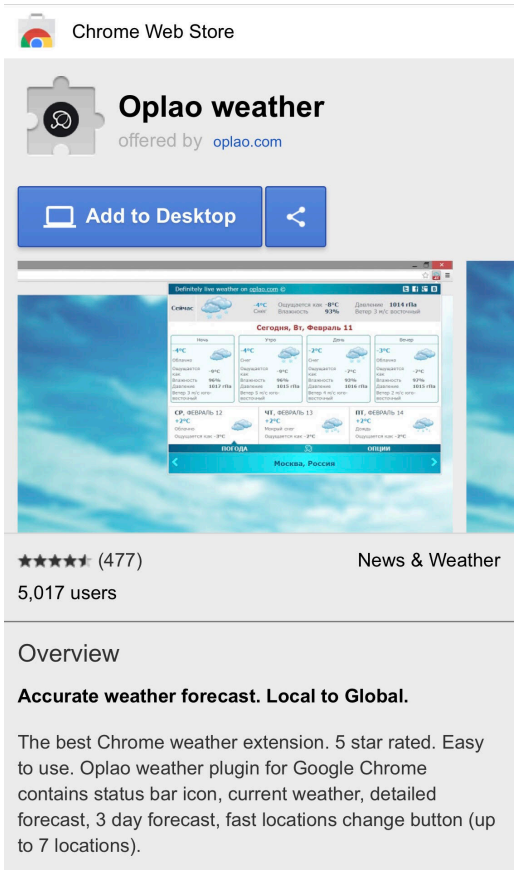

Fig. 1: The "Oplao weather" application from Chrome Extension. The application shows a correlation of its description, e.g., location weather forecast, to its request of location permission. We use this application as an example to show that knowledge can be transferred across domains.

extension, as shown in Figure 1, called "Oplao weather", is designed for accurately predicting the weather in either local or any arbitrary location specified by the user. This extension needs the location permission from user on Google Chrome to forecast the local weather.

It is worth noting that if we do not apply transfer learning to build a permission correlation system, we need extensive human work to label extensions like this in order to build a model. The reasons are two-fold. First, the extension itself does not have any descriptions related to locations, such as GPS and IP address, and therefore a direct correlation with the permission and extension description like what Whyper [37] does is not applicable here. Second, we cannot assume, like what AutoCog [40] does, that if many similar extensions on the Chrome, like weather extensions, have access to location permission, this extension needs to do so as well. The reason is that the weather extension itself does not need access to location, but an extension forecasting local weather needs. In fact, there are some extensions on Google Chrome that require users to input a city and do not have access to the location permission. To sum up, the construction of a new permission correlation system on a different platform, like

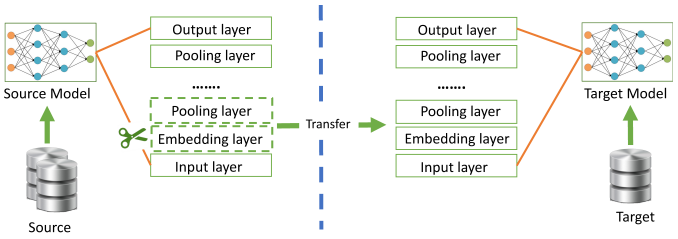

Fig. 2: An illustration of the transfer learning idea upon a neural network model. (In the figure, we transfer two particular layers, i.e., word embedding and pooling as depicted in the dashed boxes, from the source domains to the target. All the rest layers are left untouched.)

Chrome Extension, is challenging and needs human work in labeling.

We now illustrate how we transfer knowledge from the Android platform to Google Chrome and build a permission correlation system that links the extension's description with the location permission. As mentioned, there are two major types of transferred knowledge: semantics and permission. First, the extension is a weather forecast, which also exists in the Android platform. When TKPERM transfers knowledge, it transfers the semantics, such as the word "weather" related and close to "forecast" and "city" close to "local", in the word embedding space from Android to Google Chrome. Second, TKPERM transfers permission knowledge, such as the phrase "local weather" correlated to the location permission across the platform. For example, there exists an app on the Android platform, called "Garmisch-Partenkirchen", which is a tour guide of the German city. The app also mentions "local weather" and has access to the location permission.

One important task during transfers is to select corresponding domains, i.e., apps with certain permissions, on the source for the target so as to maximize the transferred knowledge. This specific transfer, i.e., from Android to the location permission on the Chrome extension, TKPERM selects three domains on the Android, which are "fine location", "coarse location" and "read contact". It is natural that both fine and coarse locations on Android are close to the target location permission on the Chrome extension. The "read contact" permission can help as well, because many apps on Android with this permission are related to event scheduling, which needs somewhat location information as well.

III. TERMINOLOGIES AND DEFINITIONS

In this section, we give a brief introduction of transfer learning and some terminologies and definitions related to transfer learning and used in TKPERM. Generally speaking, Transfer Learning (TL), as illustrated in Figure 2, is a type of Machine Learning technique that transfers knowledge from one party, called *source*, to another, called *target*, so as to improve the performance of the target. The most widely-used scenario is to transfer knowledge from a source neural network to a target by copying the weights of several layers close to the input layer. Figure 2 shows that we transfer two neural network layers, i.e., a pooling layer and an embedding, from a source to a target. Transfer Learning, in the past, has been widely used

in solving many Computer Vision (CV) and Natural Language Processing (NLP) problems [23], [42], [56].

Next, we will present several terminology and definitions of transfer learning that are used for TKPERM.

- *Domain.* A domain in transfer learning refers to a set of data, e.g., images or sentences, with similar properties. In the context of TKPERM, a domain is defined as a set of applications requesting a certain permission, particularly all the positive and negative sentences extracted from these applications and labeled by human. For example, we consider all the sentences extracted from apps requesting location permission on Android platform, which are labeled as either positive, i.e., related to location permission, or negative, i.e., unrelated to location, as a location permission domain on Android.
- *Domain Selection.* Domain selection in transfer learning refers to the procedure of selecting a set of domains to be transferred to the target. In the context of TKPERM, domain selection is that TKPERM selects a set of domains on the source platforms and transfers knowledge of these domains to a permission model on the target platform. For example, in our motivating example of Section II, TKPERM transfers knowledge from three domains, i.e., "fine location", "coarse location" and "read contact", on the Android platform to the location domain on the Chrome Extension platform.
- *Fine Tuning.* Finetuning is a procedure performed often after transfer to adjust the transferred parameters so that they can be better fit in the target domain. In the context of TKPERM, it adopts a small number of data in the target domain to adjust these parameters in the copied layers and train parameters in new layers so as to better improve the performance of the transferred model.

IV. SYSTEM DESIGN

In this section, we introduce the overall design of TKPERM, as shown in Figure 3. Generally speaking, there are four sub-procedures of TKPERM: (i) Source Domain Selection, (ii) Source Model Training, (iii) Fine Tune Data Selection, and (iv) Target Model training.

Here are the details of these four sub-procedures. First, TKPERM takes all the possible domains on the source platforms as inputs (i.e., circled one in Figure 3) and then outputs a subset of domains (i.e., circled two) for transfer. Second, TKPERM trains a source domain model (circled three) based on the selected domains. Third, TKPERM selects a small number of data in the target domain using the source model (circled four) and data on the target platform (circled five) for the purpose of fine tuning. Lastly, TKPERM builds a target model (circled eight) based on the source model (circled seven) and fine tuning data (circled six).

In the rest of this section, we introduce these four sub-procedures separately from Subsections IV-A to IV-D.

A. Source Domain Selection

TKPERM adopts a source domain selection algorithm to select the most useful source or a combination of source(s) that can help to boost the target model's performance at the target domain. While intuitively simple, the challenge of domain selection on permission-based platforms comes from the difficulty of mapping from one target domain to source domains with different permissions. That is, we followed several state-of-the-art approaches, including keyword overlapping checking, KL-divergence [27], $\mathcal{H}$-divergence [11]. Unfortunately, none of those techniques worked in our problem settings. Keywords overlap checking method as domain relevancy measurement has an as intrinsic drawback as KL-divergence. KL-divergence calculation is based on words distribution while $\mathcal{H}$-divergence takes sentences into consideration. As a result, it has the ability to capture the contextual information from the sentences. Though $\mathcal{H}$-divergence works comparatively better than others, however, it also failed to outperform our selection algorithm's performance on permission-based platforms.

Now let us introduce our domain selection, which is a greedy algorithm that identifies the best source domain(s) for Transfer Learning as listed in Algorithm 1. Here, TKPERM computes the best source(s) by evaluating the performance on the target data d_t. In each time, we select one target domain from the target platform. Then, TKPERM computes the performance of the source domain on the selected target domain using $computeallds_{f1}$ listed in Line 5 of Algorithm 1. It takes the list of the source domain,$[\mathcal{D}_S]$ and a single target domain, d_t as input. And then, TKPERM creates the set which indicates the mapping of F1 score with source domain in $[\{D_S, d_{f1}\}]$. For example, let us consider the scenario, where we want to check the overprivileged applications of "geo-location" permission from Chrome Extension platform. TKPERM computes all the nine sources with chrome's "geo-locaiton" permission and calculates F1 score for all of the source domain. For future use, TKPERM stores them in $[\{D_S, d_{f1}\}]$. After that, TKPERM selects the source which has the highest F1 score compared to others. TKPERM removes the selected source domain, d_s from $[\{D_S, d_{f1}\}]$ and aggregates it to the aggregated source list, $\mathcal{A}_\mathcal{S}$. TKPERM computes the $\mathcal{A}_\mathcal{S}$ performance on the target domain in Line 10 of Algorithm 1. In the next round, TKPERM again identifies the next best source with the highest F1 score listed in $[\{D_S, d_{f1}\}]$ and aggregates it with the previous best source list. And thus, TKPERM constructs $\mathcal{A}_\mathcal{S}$ and removes the selected source domain from $[\{D_S, d_{f1}\}]$. TKPERM repeats the whole process for aggregating source domain in $\mathcal{A}_\mathcal{S}$ (as listed between Line 7-15 of Algortihm 1), until F1 score drops below P_{best}. Otherwise, TKPERM keeps adding the next best source from $[\{D_S, d_{f1}\}]$ until there is no source left in $[\{D_S, d_{f1}\}]$.

B. Source Model Training

TKPERM trains binary classification models, particularly, Fully Connected Neural Networks (FCNN) for target permission using selected domains on the source platform. FCNN is a type of Neural Networks where each neuron in one layer has connections with all the neurons in the previous layer except the input layer. TKPERM starts the process of building our source model by combining all the data from Android platform to build word-embedding [38]. TKPERM then uses these word

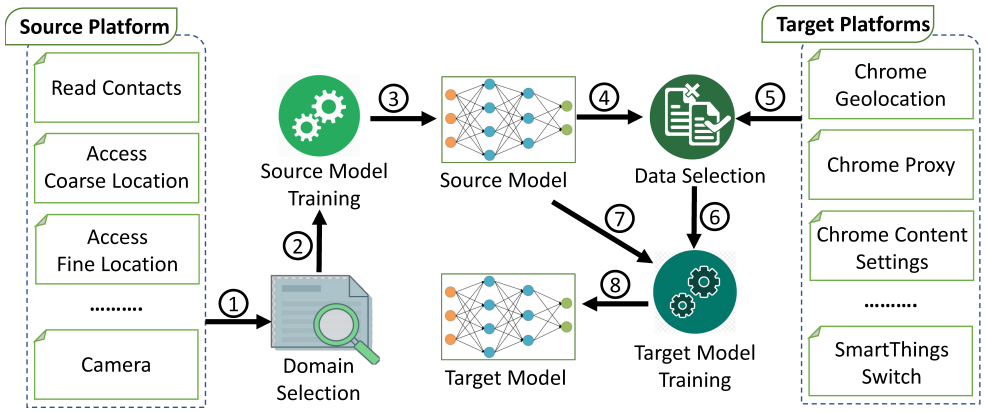

Fig. 3: System overview of TKPERM. (TKPERM has four major sub-procedures: domain selection, source model training, find-tune data selection, and target model training.)

Algorithm 1 Source Domain Selection using Greedy Selection Algorithm

Input: Source Domain Data List, $[\mathcal{D}_S]$; Target Domain Data, d_t
Output: Aggregated Source List, $[\mathcal{A}_S]$

1: **procedure** SELECTSOURCEDOMAINS
2: $[\mathcal{A}_S] \leftarrow \emptyset$
3: $P_{best} \leftarrow -\infty$
4: $P_{current} \leftarrow$ initialize to $zero$
5: $[\{D_S, d_{f1}\}] \leftarrow computealld s_{f1}([D_S], d_t)$
6: **while** $size([\{D_S, d_{f1}\}]) > 0$ **do**
7: $d_s \leftarrow highest_{f1}([\{D_S, d_{f1}\}])$
8: remove d_s from $[\{D_S, d_{f1}\}]$
9: add d_s to $[\mathcal{A}_S]$
10: $P_{current} \leftarrow computeds_{f1}([\mathcal{A}_S], d_t)$
11: **if** $P_{current} < P_{best}$ **then**
12: remove d_s from $[\mathcal{A}_S]$
13: **break**
14: **end if**
15: $P_{best} \leftarrow P_{current}$
16: **end while**
17: Return $[\mathcal{A}_S]$
18: **end procedure**

embedding vectors for building the embedding layer for all the source models. Next, TKPERM builds other layers of the neural networks, particularly, two hidden layers, which consists of 300 and 200 neurons with $ReLU$ (rectified linear unit) as the activation function. Particularly, TKPERM adopts CBOW (Continuous Bag-of-Words) encoder to translate each sentence into a vector and then trains other layers of the neural networks. Note that before using CBOW encoder, TKPERM pre-processes all the sentences by following the standard NLP practice, such as removing unicode character, punctuation, stop words (e.g., "a," "this"), and converting letters to lowercase [1]. We believe that this will help our model to learn useful features for the textual data.

We now discuss two design choices of our source model, which are CBOW encoder and FCNN. First, we choose the CBOW encoder, due to the scale limitation of our dataset for sentence encoder training. Although Existing universal sentence encoders, like InferSent [14] or SkipThoughts [26], are claimed to perform generally well, they did not outperform CBOW in our experiments. Data source mismatching and higher dimensionality of their outputs can be the possible reason [35]. Second, we choose FCNN for building our model structure for source domain knowledge distilling due to the following reason. Particularly, compared with more advanced models, such as Long Short-Term Memory, it is easier to get a better performance on a small scale dataset using FCNN model [45].

C. Data Selection

TKPERM, or in general transfer learning technique, needs a small amount of labeled data in the target domain to fine tune the transferred model to better fit in the target domain. Such data is selected by TKPERM to reduce human efforts on labeling. Specifically, TKPERM takes a source model and an unlabeled target data as inputs and then ranks all the data based on the prediction value on the listed sentences inside that application. The ranking score of an application indicates the possibility of an application whether it is worth to label or not. TKPERM selects sentences from these top-ranked applications for manual labeling.

We now describe the detailed ranking algorithm in Algorithm 2 to illustrate the process of selecting fine-tune data from the target domain. The ranking process will first divide descriptions of an application into sentences, which is shown in Line 2-3 of Algorithm 2. Then, source model starts predicting on each sentence. TKPERM increases the rank of an application by following Equation 1.

$$\mathcal{R}_{\mathcal{A}} = \sum_{j=1}^{len(\mathcal{A})} 1 \mid \lfloor \mathcal{P}(y_j|x_j) \rceil = 1 \quad \forall\, sentence, x_j \in document, \mathcal{A} \tag{1}$$

If TKPERM gets prediction result as positive for sentences inside an application, TKPERM increments the ranking score, $\mathcal{R}_{\mathcal{A}}$ of that application by one. Next, TKPERM counts ranking score ranges from zero to $\mathcal{R}_{max}$, where $\mathcal{R}_{max}$ represents

Algorithm 2 Selecting fine-tune dataset for target model using Data Selection Module.

Input: Source Model, $\mathcal{M}_{\mathcal{S}}$; Unlabeled Target Domain Dataset, $[\mathcal{A}_{\sqcup}]$
Output: Fine-tune Dataset, $[\mathcal{D}_{\mathcal{F}}]$

1: **procedure** SELECTFINETUNEDATASET
2: **for each** document, $\mathcal{A} \in [\mathcal{A}_{\sqcup}]$ **do**
3: **for each** sentence, $d_t \in \mathcal{A}$ **do**
4: $pred \leftarrow$ prediction$(d_t, \mathcal{M}_{\mathcal{S}})$
5: **if** $pred = 1$ **then**
6: $\mathcal{R}_{\mathcal{A}} \leftarrow \mathcal{R}_{\mathcal{A}} + 1$
7: **end if**
8: **end for**
9: add $\{\mathcal{A}, \mathcal{R}_{\mathcal{A}}\}$ to $[\mathcal{D}_{\mathcal{R}}]$
10: **end for**
11: $[\mathcal{D}_{\mathcal{R}}]^* \leftarrow sorted_{desc}([\mathcal{D}_{\mathcal{R}}])$
12: $[\mathcal{D}_{\mathcal{F}}] \leftarrow top_{20}([\mathcal{D}_{\mathcal{R}}]^*)$
13: Return $[\mathcal{D}_{\mathcal{F}}]$
14: **end procedure**

highest number of predicted positive sentences inside an application. Thus, TKPERM builds a list, $[\mathcal{D}_{\mathcal{R}}]$ containing a set of applications and their final rank. For example, an application where four sentences have been predicted as positive, the ranking score of that application will be four. This process is based on two intuitions: (1) selected source models have enough knowledge to distinguish data in the target domain, thus it can recognize potential positive sentences in the target domain; (2) source models are trained with many negative sentences (irrelevant to the considered permission), thus source model can successfully exclude negative sentences from target domain as well.

According to the state-of-the-art [23] transfer learning on NLP domain, we do not need large scale data to fine-tune the model. Small scale data is enough to fine-tune the target model, which shows significant improvement over the target domain. After finishing the ranking process for all the applications in a domain, TKPERM sorts the applications based on their ranking score in descending order according to Equation 2 and thus, TKPERM gets the sorted list, $[\mathcal{D}_{\mathcal{R}}]^*$.

$$[\mathcal{D}_{\mathcal{F}}] = [sort_{desc}[\{\mathcal{A}_i, \mathcal{R}_{\mathcal{A}_i}\}_{i=1}^{m}]]_{j=1}^{n} \quad (2)$$

where sorting is based on the descending order of $\mathcal{R}_{\mathcal{A}_i}$, total number of applications in target domain, m and the total number of considered applications, $n = 20$.

From the sorted list, TKPERM selects top application list $[\mathcal{D}_{\mathcal{F}}]$, which have a higher rank than others, as mentioned in Line 12 of Algorithm 2. These top applications are divided into training and validation set based on a hyperparameter.

D. Building Target Domain Model

We now describe how TKPERM builds the target domain model from two aspects: transferring and fine-tuning. First, TKPERM transfers two layers of the source domain model, i.e., one embedding layer and another pooling, from the source domains to the target. Second, TKPERM fine-tunes the transferred model based on the data selected in Section IV-C. Say, TKPERM selects data $D_T^t \cup D_T^v$ (D_T^t= training data, D_T^v= validation data) for fine-tuning process: A human needs to label all $D_T^t \cup D_T^v$.

TKPERM then try different combinations of hyperparameters in the target domain and select these hyperparameters based on the performance of the target model on D_T^v. Specifically, these hyperparameters include the number of hidden layers, different optimizer (such as Stochastic gradient descent, Adagrad, and Adam optimizer), dropout for regularization, learning rate, epoch, pooling type, and freezing model's lower layer. TKPERM enumerates all the different combinations of the hyperparameters and chooses the one with the best performance on D_T^v for the selected target domain.

V. IMPLEMENTATION

In this section, we first describe all the datasets used in the implementation and evaluation of TKPERM in Section V-A, and then present the models and hyperparameters built or selected by TKPERM in Section V-B.

A. Datasets

We now describe the collection procedure, manual labeling, and then some statistics of the datasets.

1) Collection Procedure: We collect datasets from four different platforms, including Android, Chrome Extension, IFTTT, and SmartThings.

- *Android.* Android is a mobile platform that provides range of applications available in the Play Store. We adopted the crawled data, provided by the authors of Autocog [40], in order to be consistent with prior works. This dataset contains the descriptions along with the permission information of 45,811 android applications that were available in May 2014 in Google Play Store.
- Chrome Extension. Chrome Extension is a platform that provides small programs integrated in Chrome browser and built in HTML, JavaScript, and CSS to enhance the browsing experience. We collected 1,059 chrome extension applications in November, 2018. At that time, there were twelve different categories available in Chrome store, including accessibility, blogging, by google, news, shopping, web development, fun, photos, productivity, search tools, communication, and sports. We build a Chrome data crawler to get all the application's information. First, we extract all the IDs of chrome extension applications by browsing webstore. Then, we made a query with the id in "chrome extension source viewer" [2]. Finally, with the help of our crawler, we were able to get the detailed information, such as title, description and required permissions. Then we stored those information and constructed our chrome extension dataset.
- IFTTT. IFTTT is a platform that supports trigger-action programming (TAP) with IoT automations. We collected 259,523 IFTTT recipes in October 2017 using our crawler built with python and beautiful soup. Here are the detailed procedures. We searched the recipes for each of the services by going through each service page in IFTTT. For each service, we collected the relative information including its description, title, trigger title, trigger service, action title, action service, and url (i.e., link to the recipe). One recipe

may appear both in trigger service page and action service page. For removing redundancy, we did not collect the recipe twice, and each recipe has two permissions, one for trigger service and the other for action service.

- SmartThings. SmartThings is another popular platform in IoT domain provided by Samsung. We collected 243 SmartThings applications in August 2019. Here is how we collect them. We build a python crawler to extract data from the official Github repository[1] of SmartThings. Here, we considered the capability for each application as its permission. At the time, there were 39 different capabilities, including lock, switch, motion sensor, acceleration sensor, presence sensor, and contact sensor. We select the top three capabilities based on the size of each dataset.

2) Manual Labeling: After data collection, we choose top, common permissions, i.e., these with enough applications on each platform, and manually labels sentences from descriptions of applications with these permissions. A detailed list of these top permissions is shown in the second column of Table I. We select all the sentences on IFTTT, SmartThings and Chrome extensions, i.e., our target platform, for human labeling; We only select these applications that are classified as positive by AutoCog for human labeling in the Android platform due to the large number of involved applications.

Our labeling process is as follows. We first present the official documents of our platforms and permissions to two students, who later become co-authors of the paper, as annotators and also ask them to get familiar with all the platforms under our study. We then ask them to label some dataset with our human annotated ground truth to ensure that they understand these platforms and permissions. Next, we ask them to label whether a sentence in a particular application is related to a certain permission requested by the application. If the student considers the sentence indicates the requirement of a permission, he or she will label it as positive; otherwise, negative.

Once we have labels from two annotators, we will integrate all the labels from these two sources together. Interestingly, several sentences were vague enough to make a confusion among the annotators, as a result disagreement occurs. On an average we have a very good consistent labels between two annotators, i.e., we have an agreement rate as 97.89%, and kappa as 0.901. That is, we believe that our dataset is with a high quality based on prior works [22], [50]. Furthermore, for these disagreements, we resolved the conflict issue by presenting the case to these two annotators again and asking them to reach a consensus. They reached a consensus for all the conflicts after a discussion.

We now describe a concrete example in which two annotators have disagreement but reached a consensus after a thorough discussion. Here is the sentence: "When you have a meeting, auto create a note at Evernote", which belongs to an IFTTT recipe requiring access to Google Calendar. Two annotators have disagreement because one thinks that this sentence has no relationship with Google Calendar, while the other thinks that a recipe can only know that you have a meeting based on an access to Google Calendar. After a discussion, they agree that the latter annotator is correct as the IFTTT platform does not provide other permissions so that a recipe can know that the user has a meeting.

[1]https://github.com/SmartThingsCommunity/SmartThingsPublic/tree/master/smartapps

TABLE I: Manually-labeled Data distribution of 20 different domains from Android, IFTTT, Chrome Extension and SmartThings. Android is the source platform and the rest are target platforms.

Plat.	Permission	#Sent.	#Pos. Sent.	#Doc.	#Pos. Doc.
Android	Fine Loc.	16,402	728 (4.44%)	635	635 (100%)
	Coarse Loc.	5,550	208 (3.75%)	193	193 (100%)
	Camera	498	166 (33.33%)	11	11 (100%)
	Read Cal.	802	401 (50.00%)	16	16 (100%)
	Read Con.	842	421 (50.00%)	17	17 (100%)
	Record Au.	366	183 (50.00%)	10	10 (100%)
	Wr. Settings	1,524	398 (26.12%)	31	31 (100%)
	Send SMS	8,398	407 (4.85%)	286	286 (100%)
	Write APN	1,811	92 (5.10%)	35	35 (100%)
IFTTT	Evernote	202	133 (65.84%)	145	85 (58.6%)
	BMW Lab	77	52 (67.53%)	65	43 (66.2%)
	Facebook	158	84 (53.16%)	115	45 (39.1%)
	G. Cal.	144	88 (61.11%)	102	73 (71.6%)
	G. Con.	85	50 (58.82%)	49	43 (87.8%)
Chrm.	Geoloc.	1,540	126 (8.18%)	138	67 (48.6%)
	Proxy	2,391	483 (20.20%)	123	98 (79.7%)
	C. Settings	774	92 (11.89%)	58	28 (48.3%)
Smart Things	Lock	34	10 (29.31%)	30	8 (26.67%)
	Motion	73	40 (54.79%)	60	35 (58.33%)
	Switch	185	118 (63.78%)	153	111(72.55%)

3) Statistics: We now describe the statistics of our datasets after human labeling. In total, we labeled 36,193 sentences from 1,234 Android applications, 666 sentences from 476 IFTTT recipes, 4,705 sentences from 319 Chrome extensions and 292 sentences from 243 SmartThings applications. Table I shows the number of total and positive sentences and documents in each platform. Note that a document means all the descriptions from a certain application on a platform.

Now let us describe several observations made from our manually-labeled dataset. First, the number of positive sentences on Android and Chrome Extension is scarce when compared with negative ones. On the Android platform, only 8.3% of sentences, i.e., 3,004 out of 36,193 labeled ones, are positive, i.e., indicating the selected permission. The scenario on Chrome Extension is similar, with 13.42% positive sentences. Such a percentage is much higher on SmartThings and IFTTT, which are 49.29% and 61.29% respectively.

The reason of such a drastic difference on different platform is due to the length of descriptions. Android apps usually have the longest descriptions on many functionalities that are unrelated to the app's permission request and the same applies to Chrome extensions. The descriptions of IFTTT recipes and SmartThings applications are usually short, which precisely presents the permission requirement. Therefore, the percentage of positive sentences on Android and Chrome extensions is scarce, but the one on IFTTT and SmartThings is abundant.

Second, many applications on IFTTT, SmartThings and Chrome extensions are overprivileged as indicated in the percentage of positive documents on these platforms. Such an observation emphasizes our motivation in building a permission correlation system on these platforms. Note that all the documents on Android are positive because we only select

applications that are labeled as positive by AutoCog, a prior permission correlations sytem.

Lastly, although the percentage of positive sentences on Android is small, the absolute number is still large. The reason is that Android platform is with much more applications and users as compared with others. That is also the major reason that we adopt the Android platform as the source platform for TKPERM.

B. Models and Hyperparameters

In this section, we describe our source domain model that is used for transfer and also several hyperparameters. Note that we used the Amazon Elastic Compute Cloud (EC2) resources to run all of our experiments. The instance we used is called 'p3.2xlarge' with one NVIDIA Tesla V100 GPU, 16 Gibibyte GPU memory, 8 virtual central processing units (vCPUS) and 61 Gibibyte Main Memory. The operating system of this instance is the 'Deep Learning Amazon Linux Version 23.0'.

1) Source Domain Model: Our source domain model is built on our manually labeled sentences on the Android domain. One major challenge comes from the scarcity of positive sentences on the Android platform: if we randomly select data from all sentences for training, tuning, and evaluation, there is a chance of source domain becoming fully biased to negative sentences [10], [30]. Therefore, we need to fix the positive and negative data ratio and train source domain model. With a fixed ratio, we randomly sampled 70% data to train the model, 10% to tune the hyperparameters, and rest 20% data to evaluate the model's performance. Finally, our results show that a 1:1 ratio will produce the best results on our evaluation set.

Table II shows the performance, including Accuracy, Precision, Recall and F1 score, of the source model on each of the nine different source domains. The average source model performance in terms of F1 score is 84.2%. We calculate all the numbers based on the following criteria. The source domain model first extracts sentences from each of the document and starts classification on those sentence data. After that, the model predicts whether an individual sentence belongs to positive class (i.e. 1) or negative class (i.e. 0). We will consider the document, i.e., an application, as positive, if one of each sentences is considered as positive by the source domain model. If none of a document's sentences are positive, we will consider it as negative.

One interesting observation for our source domain model is that permissions with straightforward descriptions are with the highest F1 score. For example, the F1 score for "Send SMS" is as high as 97%, because the descriptions of this permission usually involves a direct description of SMS. To the contrary, the F1 score for "Coarse Location" is the lowest, because the request for "Coarse Location" is often vague with the request for "Fine Location". Additionally, there are many ways to describe the request of locations, such as local weather as we described in the motivating example.

2) Hyperparameter Selection: Hyperparameter selection in TKPERM is automatic because TKPERM reserves 10% of the total data as a validation set so that TKPERM can select the best hyperparameter. Specifically, TKPERM considers the following hyperparameters for training the source model,

TABLE II: Performance, i.e., accuracy, precision, recall, and F1 score, of source models on different source domains in Android platform.

	Performance			
Permission	**Acc.**	**Prec.**	**Rec.**	**F1**
Fine Location	85%	73%	84%	78%
Coarse Location	84%	53%	84%	65%
Camera	88%	80%	89%	85%
Read Calendar	89%	87%	89%	88%
Read Contact	92%	92%	90%	91%
Record Audio	84%	83%	83%	83%
Write Settings	87%	69%	86%	77%
Send SMS	93%	93%	100%	97%
Write APN	92%	88%	97%	94%
Total	88.22%	79.78%	89.11%	84.20%

which are epoch, batch size, learning rate, dropout, pooling types, word vector dimension, gradient norm limit, number of selected fine tuning data, and number of hidden layers. Many of these hyper-parameters change according to the different sources and target models.

We now describe some hyperparameters that are consistently chosen by TKPERM. Specifically, TKPERM selects $lr = 0.01, b = 256, e = 20$, where lr = learning rate, b = number of batch size and e = number of epoch, as these settings produces a good source model for all the domains. TKPERM also selects the dimension of this word embedding to 300 as this dimension works comparatively better than other dimensions. During the ranking algorithm, i.e., Algorithm 2, from the sorted list, TKPERM selects top 20 application list $[\mathcal{D}_{\mathcal{F}}]$, which have the higher rank than others, as mentioned in Line 12. Among top 20 applications, 15 of them are considered in training set, while the rest five applications are selected as validation set for tuning hyperparameters.

VI. EVALUATION

In the following section, we present the evaluations of TKPERM. To check if it is an effective and efficient system for detecting overprivilege issues, we evaluate both the performance and the computation overhead of TKPERM. For the performance evaluation, we report the end-to-end performance for detecting overpriviledged applications, as well as the performance of different components in TKPERM. We show that TKPERM is effective for overprivilege detections on all the three platforms we test (average F1 score is 90.02% with 988 pieces of labeled data). Besides, we demonstrate that our greedy domain selection algorithm outperforms the popular source domain selection approaches such as $\mathcal{H}$-divergence for transfer learning. TKPERM also reduces the labeling cost by selecting potential useful data for the fine-tuning purpose. Our experiment results indicate the performance improvement of the data selection approach than randomly picking fine-tune data. In the end, we also evaluate the performance overhead of TKPERM, showing that TKPERM is scalable for a large number of applications.

A. Evaluation Questions and Metrics

To evaluate the performance of TKPERM, we want to answer the following questions:

TABLE III: Detailed Performance of TKPERM in different target domains.

Plat.	Permission	Performance			
		Acc.	Prec.	Rec.	F1
IFTTT	**Evernote**	84.6%	77.53 %	89.61%	83.13%
	BMW Lab	94.00%	99.99%	90.90%	95.24%
	Facebook	90.00%	78.72%	100%	88.09%
	G. Cal.	88.51%	86.96%	98.36%	94.30%
	G. Con.	94.11%	93.33%	100%	98.41%
Chrm.	**Geoloc.**	89.43%	85.96%	90.74%	88.29%
	Proxy	89.81%	89.24%	98.80%	93.78%
	C. Settings	76.74%	68.97%	95.24%	85.31%
Smart Things	**Lock**	93.33%	75.00%	100 %	85.71%
	Motion	82.22%	77.14%	100%	87.10%
	Switch	91.36%	89.38%	100%	94.39%

- What is the end-to-end performance of TKPERM?
- What is the performance of each component in TKPERM?
- What is the computation overhead of TKPERM?

Answering the first two questions helps to show the effectiveness of TKPERM, as well as the effectiveness of design decisions in each component. Answering the third question helps to understand the scalability of our solution. For evaluating the performance of TKPERM and its components, we use F1 score, which is the harmonic mean of precision (the fraction of relevant instances among the retrieved instances) and recall (fraction of relevant instances that have been retrieved over the total amount of relevant instances). Higher precision will ensure the low FP (False Positive), whereas, a higher recall will ensure the low FN (False Negative). As a result, higher F1 score will provide both low FP and FN. That's why for evaluating the model's performance, we are selecting F1 score as the primary measurement criteria. Models with higher F1 score will have a good chance of identifying overprivileged applications.

B. Overall performance of TKPERM

As mentioned in Section V-A, we collect data from three popular platforms with third-party applications (IFTTT, Chrome extension, and SmartThings). To evaluate the overall performance of TKPERM, we experiment with 11 popular and sensitive target permission domains on the three different platforms. First, we use a small portion of data from the target domain to fine-tune the target model. According to the state-of-the-art approaches, we do not need to have a large amount of fine-tune data [23] to get good performances in the target model. We extract the fine-tune data systematically as described in Section IV-C to reduce the labeling cost and ensure the quality of fine-tuning data. Table I depicts the data distribution of all 11 different target domains from three different platforms. While tuning the hyperparameter, we validated the model's performance on the validation dataset. We select hyperparameter automatically from the list of hyperparameters as described in Section IV-D.

Comparison with baseline. To get more insight on the performance of the transfer learning, we evaluate our approach comparing with "No Transfer" technique (considered as baseline). To have a fair comparison with "No Transfer" scenario, we use the same amount of labeled data to train the model directly. Table IV demonstrates the contrast of the performance between transfer with no transfer approach. By observing the improvement ratio for all target domains, we find that TKPERM outperforms the baseline approach in every target domain. On average, it exceeds the baseline by 12.62%.

Measurement of overprivilege apps. We find that the app overprivilege is a pervasive issue. On average, we find 32.33% of apps are overprivileged. 135 apps (28.36%) from IFTTT, 114 apps (35.73%) from Chrome Extension, and 80 apps (32.9%) from SmartThings are overprivileged.

In the following, we report the component-level performance of TKPERM. In particular, we highlight the performance of the source domain selection module and data selection module because we make unique design choices for these two components.

C. Source domain selection module performance

To successfully transfer knowledge to a target domain, we need to select the best source domain(s) for each given target domain. In transfer learning, selecting the best source domain is a challenging problem [8], [53]. In addition, in our problem settings, permissions in one platform cannot be easily mapped to permissions of other platforms intuitively due to the diversity of platforms. Moreover, if source domain contains unnecessary data (i.e., inappropriate for completing the targeted task), then our target model may experience "negative transfer" [43]. TKPERM overcomes these challenges by proposing the greedy selection based domain selection algorithm (listed in Algorithm 1).

To compare our greedy selection algorithm's performance on selecting the best source domains, we also ran experiments with the state-of-art domain selection algorithm $\mathcal{H}$-divergence algorithm [11] to find the most relevant source domain. Process of using $\mathcal{H}$-divergence algorithm to select the best source domain includes a binary classification problem, where the source and target domain has two different labels (e.g., source data is labeled as 1, while the target data is labeled as 0). The intuition is that if the classifier can hardly distinguish two datasets, e.g., making a lot of errors during the evaluation, then these two domains are very relevant. For source-target domain combination with bigger error, most likely those two domains are relevant to each other.

Table V shows the performance comparison of our greedy selection algorithm with $\mathcal{H}$-divergence algorithm to find the best source domain. We can observe that the greedy selection algorithm achieves 4.59% improvement of F1 score compare to $\mathcal{H}$-divergence algorithm. Thus, we can conclude that TKPERM can extract the best source domain for transferring the knowledge to the target domain.

One interesting phenomenon we observe is those source domains performing well in the source platform are likely to trigger good performance of their target domains in the target platform. For example, as is shown in Table II & III, "Send SMS" achieves 97% F1 score in source domain, then the performance at its target domain "BMW Lab" is also outstanding. Our greedy selection algorithm for domain selection can also identify these kinds of source domains at an early stage. In the first round of Greedy Selection

TABLE IV: Performance improvement (based on F1 score) analysis of TKPERM compared with "No Transfer" (baseline) in 11 different target domains with a highest improvement of 33.77% using transfer learning.

Plat.	Target Domain	Source Domain	Trans.	No Trans.	Improve.
IFTTT	Evernote	Coarse Location + Fine Location + Camera	83.13%	79.78%	3.35%
	BMW Lab	Send SMS + Record Audio	95.24%	85.71%	9.53%
	Facebook	Camera	88.09%	75.00%	13.09%
	Google Calendar	Read Calendar + Coarse Location	94.30%	83.54%	10.76%
	Google Contact	Read Contacts	98.41%	97.22%	1.19%
Chrome	Geolocation	Fine Location + Coarse Location + Read Contact	88.29%	62.50%	25.79%
	Proxy	Send SMS + Fine Location	93.78%	89.69%	4.09%
	Content Settings	Fine Location + Read Contact	85.31%	59.61%	25.7%
Smart Things	Lock	Write Setting	85.71%	75.00%	10.71%
	Motion Sensor	Read Contact	87.10%	53.33%	33.77%
	Switch	Send SMS + Read Calendar	94.39%	90.09%	4.3%

TABLE V: Performance comparison of $\mathcal{H}$-divergence with Greedy Selection algorithm for selecting best source domain for transferring learning using TKPERM in IFTTT platform.

Target Domain	Src Selection	Src Domain(s)	F1
Evernote	$\mathcal{H}$-divergence	Read Calendar	75.86%
	Greedy Select.	Coarse Location + Fine Location + Camera	83.13%
BMW Lab	$\mathcal{H}$-divergence	Read Contact	92.3%
	Greedy Select.	Send SMS + Record Audio	95.24%
Facebook	$\mathcal{H}$-divergence	Read Calendar	76.09%
	Greedy Select.	Camera	88.09%
Google Calendar	$\mathcal{H}$-divergence	Read Calendar	91.30%
	Greedy Select.	Read Calendar + Coarse Location	92.30%
Google Contact	$\mathcal{H}$-divergence	Read Contacts	99.20%
	Greedy Select.	Read Contacts	99.20%

approach, "Send SMS" achieves 93.15% F1 score which is 7.44% improvement from "No Transfer".

In addition, we check if our domain selection algorithm will cause negative transfer issues. As is shown in Table III, TKPERM provides the best target model for each of the domain. In IFTTT, we achieve the best performance with 98.41% F1 score in Google Contact domain whereas, the lowest performance is 83.13% in IFTTT Evernote domain. Interestingly, we can notice that all of the target domains achieve more than 80% F1 score; the average performance is 90.02%. Therefore, TKPERM's domain selection algorithm avoids "negative transfer" issue.

D. Data selection module performance

To further reduce human labeling effort, we design our data selection module to select high-quality data from the target domain. Here, by "high-quality data" we meant the data that is helpful for fine-tuning the target model. We described our data selection module in Section IV-C, which ranked documents from the target domain. Through our data selection process, we select the top 20 applications from the target domain. After extracting those highly ranked applications, we labeled sentences from each of the application manually. Section V-A2 describes our detailed methodology of data labeling. Table VII illustrates the effectiveness of data selection module. We can observe the comparison of performance between with and without data selection technique in Chrome Extension platform. While experimenting without data selection approach, we select randomly 20 fine-tuning applications from the target domain. To have a fair comparison, we keep the rest of the techniques the same and only change the data selection process.

We find that the data selection algorithm we proposed is effective for building models for target domains. For example, we show the comparisons of performance with our proposed data selection algorithm, and with the random data selection approach in Table VII. In the Chrome platform, we achieve significant improvement by applying our proposed data selection approach. On average, we achieve 89.13% F1 score using data selection approach while only 84.36% F1 score without data selection approach. This indicates 4.77% improvement using data selection approach in the Chrome platform. In addition, we can observe similar improvements (0.75% on IFTTT and 4.45% on SmartThings), as is shown in Table VI.

TABLE VI: Performance (F1 Score) comparison among three settings- no transfer, without data selection, and data selection in three different target platforms and comparison of performance improvement compared to "No Transfer" (baseline).

Plat.	Perform.	Configuration		
		No Trans.	W/ DS	With DS
IFTTT	F1 score	84.25%	91.08%	91.834%
	Improv.	-	6.83%	7.584%
Chrome	F1 score	70.6%	84.36%	89.13 %
	Improv.	-	13.76%	18.53%
Smart Things	F1 score	72.80%	84.65%	89.1%
	Improv.	-	11.85%	16.3%

E. Computation Overhead

To evaluate the scalability of the TKPERM, we measure the computation overhead for the system. As mentioned in Section V-B, we use Amazon Elastic Compute Cloud (EC2) resources and NVIDIA Tesla V100 GPU to train our model. We run each experiment for 11 times to compute the average computation overhead. First, we need to run the experiment on nine different source domains for any given target domain, then use the greedy algorithm we have showed in Algorithm 1 to get the best combination of source domain. We have reached on

TABLE VII: Performance improvement analysis of fine-tune data selection process of two different techniques- with and without data selection (DS) module in Chrome extension platform. The results clearly show that DS improves the F1 score of the transfer learning on all three target domains.

Target	Source	Perf.	W/DS	DS
Geoloc.	Fine Location + Coarse Location + Read Contact	F1	83.10%	88.29%
		Improv.	-	5.19%
Proxy	Send SMS + Fine Location	F1	93.61%	93.78%
		Improv.	-	0.17%
C. Sett.	Fine Location + Read Contact	F1	76.36%	85.31%
		Improv.	-	8.95%

average of 90.02% F1 score on target domain with an average size of 94 documents.

Table VIII shows the performance overhead of running experiments on different permissions and platforms, which includes the information about the size of the data (source data + target data) and the time cost of TKPERM (getting the result on the best source combination).

Note that the performance overhead of TKPERM is a one-time cost, once we train a model for one target permission, we do not need to retrain the model for new applications. Therefore, based on the data from Table VIII, TKPERM is scalable.

F. Factors impacting transfer learning

By looking into the data sets in original domains and target domains, we have the impression that the word embedding and permission knowledge might be the major knowledge that gets transferred for TKPERM. We design some experiments to control these two factors to give more insights on why transfer learning works in TKPERM. These experiment results match our intuition. Indeed, word embedding and permission knowledge have significant impacts on the target model performance.

1) Effect of word Embedding: TKPERM uses source domain's (Android) word embedding and transfers it to the target domain. As the description in Android is both rich and well structured, our source model initialized with useful features. By transferring the pre-trained word embedding to the target domain, it also shows improvement over the target model.

To investigate the impact of word embedding, we ran several experiments by using word embedding built from source platform and target platform. Our intention was to check whether model initialization phrase is getting any help from the source data or not. Table IX illustrates the performance comparison between word embedding from source data and target data. The key difference between these two settings is the selection of platform while building the word embedding. For checking the performance of source word embedding, we directly use the embedding built from source data (i.e. Android). Whereas, to test the performance of word embedding of the target data, we use the corresponding target platform's full data to train the word embedding. Important thing to note that, training word-embedding is an unsupervised approach. We do not need the labeled data for that. So, we use the large corpus data from the target platform to built the word embedding for all the target domains of that platform. For example, to experiment with "Geolocation" permission in Chrome Extension platform, we used all the description (from 1,059 applications) to train Chrome word-embedding, and then we used that in target model's embedding layer.

Results in Table IX highlight the impact of source platforms word embedding. Indeed, TKPERM gets benefited by using source platforms word embedding with an improvement of 3.69% F1 score (average) in all three target platforms. Finally, we can conclude that we do not need to train different word embedding for different target platforms. It reduces the training cost for target platforms.

2) Knowledge of Permissions: TKPERM learns specific features from the textual data for a particular permission. Then, it maps such features to the corresponding permission. In particular, TKPERM correlates requested permission with the description, and thus resulted output indicates whether a particular application is overprivileged or not. In our investigation, we observed that several permissions (though the applicability may differ with each other) overlap with each other in source and target platform. For example, Geolocation from Chrome, and Fine Location & Coarse Location from Android; Google Calendar from IFTTT, and Read Calendar from Android, Google Contact from IFTTT, and Read Contact from Android— all of these pair of permissions from two different platforms, descriptions of which have some common characterestis because of the similar functionality. In our experiment (as illustrated in Table IV), we noticed the improved performance of such correlated domains. Finally, we can conclude that, permission knowledge from source domain is also driving the performance of transfer learning.

VII. Case Study

TKPERM identifies 329 overprivileged applications from all the different platforms. We have responsibly reported these applications to their developers. In the following, we will show a few examples of overprivileged applications from each of the three platforms, such as *Tomorrow's Forecast on the way home* and *YSA email automation* from IFTTT, *GamingHub* and *Private Internet Access* from Chrome Extension, *Button Controller* and *Hue Mood Lighting* from SmartThings.

A. IFTTT

"Tomorrow's Forecast on the way home" is an IFTTT recipe that helps a driver to get the upcoming weather forecast while driving back to home from work. To install this recipe, the user needs to give access to her BMW car. Then, she will be notified about the upcoming weather forecast during a particular time of the day. They describe their recipe's functionality as "Connect the weather service before running." Just by reading the title or description, the user will not expect that this recipe would access her BMW car because the recipe does not mention anything relevant in its description and title.

"YSA email automation" is another overprivileged recipe from the IFTTT platform. This recipe accesses the sensitive Google contact data, which is not relevant to its description "YSA email automation New member email tree". After installing this recipe, whenever a user adds a new google contact, the recipe will send out an email notification. Users would not have expected the app to access their Google contact information when they install the recipe.

TABLE VIII: End-To-End evaluation for the overall computation cost of TKPERM. Note that the computation cost include source model training time and 29 iterations of transfer learning

Plat.	Target	Source	#Doc. in Target	#Doc. in Source	Time (hh:mm:ss)
IFTTT	Evernote	Coarse Location + Fine Location + Camera	145	839	33:27:03
	BMW Lab	Send SMS + Record Audio	65	296	14:08:40
	Facebook	Camera	115	11	22:57:20
	Google Calendar	Read Calendar + Coarse Location	102	207	15:15:18
	Google Contact	Read Contacts	49	17	18:40:17
Chrm.	Geolocation	Fine Location + Coarse Location + Read Contact	138	845	07:37:28
	Proxy	Send SMS + Fine Location	123	921	06:54:01
	Content Settings	Fine Location + Read Contact	58	652	09:42:45
Smart Things	Lock	Write Setting	30	31	03:47:59
	Motion Sensor	Read Contact	60	17	04:09:44
	Switch	Send SMS + Read Calendar	153	302	14:11:08

TABLE IX: Performance comparison of two different word embeddings (one is built from source platform, while the other the one is from target platform) while training the target model.

Plat.	Perform.	Word Embedding	
		Target	Source
IFTTT	F1 score	86%	91.83%
	Improv.	-	5.83%
Chrome	F1 score	88%	89.13%
	Improv.	-	1.13%
Smart Things	F1 score	85%	89.1%
	Improv.	-	4.1%

B. Chrome Extension

"Private Internet Access" is a popular extension on the Chrome platform with more than 170,000 users. It helps to encrypt user network traffic and keeps them protected while connected to the internet. They protect the user by webRTC (real-time web communication) blocking, sorting the gateways by latency. To install this extension in the Chrome browser, user needs to allow the Content Settings permission, which is the ability to "Change your settings that control websites' access to features such as cookies, JavaScript, plugins, geolocation, microphone, camera, etc." according to the official document. TKPERM detects this extension as overprivileged as its access to sensitive data like cookies, geolocation, microphone, and camera without clear justification. We have received some initial, relatively positive feedbacks from this extension's developers—they partially acknowledged our issue and we are still in the process of talking with them regarding a possible solution.

"GamingHub" is another popular extension on Chrome platform with over 20,000 users. It enables user quick & elegant access to some of the most popular web games to date. It does so by displaying them as quick access links on browser's *New Tab* Page. TKPERM detects this app as overprivileged because of its access to user's physical location without any links to its described functionality of displaying web games. It is worth noting that many other permissions, such as "Read and change all your data on the websites you visit" and "Read and change your browsing history", of the extension, are also overprivileged.

C. SmartThings

"Button Controller" is a smartapps from SmartThings platform. The functionality of this app is to "control devices with buttons like the Aeon Labs Minimote". This description reflects that it will operate like a switch that can control devices by turning on/off. Unfortunately, there does not exist anything, neither in their title nor in description, mentioning that they will also access the "Lock" permission. Interestingly, if a user allows all the permissions of the app, which are switch, lock, music player, and alarm, she grants access to the app in both locking and unlocking her door, which is fearsome.

"Hue Mood Lighting" is another overprivileged application from SmartThings platform. From the description listed as "Sets the colors and brightness level of your Philips Hue lights to match your mood", it is clear that this application will change the color and brightness of lights. TKPERM detects the application as overprivileged as it is requesting permissions for accessing motion sensor without any clear justification. It is worth noting that the application also requests many other unnecessary permissions, such as contact sensor, acceleration sensor, switch, presence sensor, smoke detector, water sensor, and button, posing even a greater threat.

VIII. Discussions & Limitations

When choosing targeted platforms, we selected popular platforms with third-party applications because the overprivileged problems in these platforms would have more impacts on the users. Currently, two of our selected target platforms (IFTTT and SmartThings) are both in the domain of IoT. However, we have Chrome Extension as another target platform which works very differently comparing to these IoT platforms and the original domain Android platforms. This justifies that TKPERM does not just work for to mobile and IoT spaces. In the future, we plan to include more target platforms such as - SDN Controller, VR (Virtual Reality) applications when these platforms gain more popularity and have more third-party applications.

On one platform during our experiments, we studied some of the most popular and security-sensitive domains that are highly representative to the platforms like IFTTT, Chrome Extension, and SmartThings. The rest of the domains are either not very important or lack enough data.

Our current detection of overprivilege is based on the mismatching of privacy implications in apps' descriptions and their permission requests. The reason is that, if apps request permissions irrelevant to the functionality described in their descriptions, they might get access to information which users did not expect them to access. We assume that users will read an apps' titles and descriptions to make the app installation decision. It is an another interesting topic to study users' decision making processes for app installations. Besides, the scope of the paper is to check the unexpected permission requests that are not associated with apps' functionalities. It is another interesting research question to check further how apps use these unexpected permissions. For example, when source code or binary of an app is available, static and dynamic analysis of the app can help understand how the app is managing data and controlling other devices.

One interesting property of TKPERM is that our model, unlike some prior work [51], still has a high accuracy on applications with short descriptions. For example, TKPERM can correctly correlate short sentences with permissions, such as "Turn cameras on when I am away", "Add Google Contacts to a Spreadsheet", and "Weather for your location". This phenomenon is especially prominent on IFTTT platform as almost all the descriptions are very short compared to those on other platforms. We believe that there are two reasons. First, our model is transferred from the Android platform, which has abundant training data. Second, most short sentences are concise, i.e., with enough information for the correlation purpose.

IX. Related Work

In the following, we first describe previous studies of applying transfer learning at the natural language processing and computer vision fields in Section IX-A. Next, we present existing studies on permission-based access control, such as these using program analysis and natural language processing, in Section IX-B.

A. Transfer Learning

In Computer Vision, transfer learning is widely used from general to task-specific cases [55]. In addition, most of the works achieve good performance over the target model just by transferring the first few layers of the source model [29], [46]. Recently, researchers working with language modeling, sentiment analysis, question-answer modeling are adopting transfer learning in NLP domain [23], [39], [42], [56]. There are several ways to do the transferring process, including-reusing instances from source [13], multi-task learning [25]. A commonly used approach is to pretrain embeddings that capture the contextual information which is used as different features or with the intermediate layer's input [12], [15], [38].

Following the same criteria, we consider our word embedding reusing process as part of the *Transfer Learning* methodology. In contrast to the existing transfer learning work in NLP domain, none of the previous work captures the security and privacy context from the textual data. TKPERM extracts the security and privacy knowledge from one domain and helps to identify the overprivileged applications in a new target domain with a small scale fine-tuning data.

B. Permission Based Access Control

In mobile and IoT platforms, permission-based access system has received a lot of attention by the research community [9], [17]–[19], [32], [48], [54]. Rahmati et al. explored the benefit of context-specific access control in Android platform [41]. Whereas, Acar et al. investigated the failure of the current concept of permission granting system [7]. Based on user study on Android platform, Backes et al. prioritized contextual integrity for the design of the permission system [52]. Researchers have also investigated overprivileged application in IoT platforms [24] from both programming and description analysis perspective. Fernandes et al. built a static analysis tool to evaluate overprivileged application [20].

In the access control area, the closet line of work to TKPERM is NLP-based analysis to detect overprivilege issues by analyzing descriptions. Previous works have used NLP techniques to conduct privacy & security analysis under different situations such as mobile apps [31], [36], malware [47], privacy policy [28], [44], vulnerability in IoT applications [21], [33]. For mobile platform, WHYPER [37] and AutoCog [40] also addressed this overprivileged problem. However, to protect the user from the overprivileged application, it becomes comparatively difficult for IoT platform compared to the mobile platform due to the privacy implications of physical context in IoT platforms. Tian et al. redefined the overprivilege to be more user-centric and captured the corresponding problem in SmartThing platform [49].

To the best of our knowledge, most of the previous research in this area only study one or two systems, which cannot be easily extended to other platforms. In comparison, TKPERM is a general framework for detecting app overprivilege across many different platforms, including- mobile, IoT, and web browser spaces. Thus, we needed to extract common characteristics among all the overprivileged applications from all the different platforms. Compared to previous papers, TKPERM enables using the existing knowledge to identify the overprivilege problems in different platforms with small scale data.

X. Conclusion

Permission-based platforms are popular and diversified, ranging from Android to Chrome extensions to IFTTT. One critical yet open task on these permission-based platforms is to detect overprivileged third-party applications, which have access to private user data. Prior works have studied and designed permission correlation system that connects permission with third-party applications' descriptions. However, all the prior works can only be used on a certain platform, i.e., the emergence of a new platform requires tedious and expensive efforts to have high-quality and large amount of labeled data. Moreover, it will require corresponding manual efforts in building and tuning models even if the data is labeled.

In this paper, inspired by the success of applications of transfer learning on images and natural language processing fields, we design TKPERM, the first transfer-learning system that transfers the permission knowledge from one platform to another. Our insight here is that these platforms, though being different, are user-facing and share many common knowledges, such as permissions and semantics.

We have implemented a prototype of TKPERM that transfers knowledge on the Android platform to three different platforms, which are IFTTT, Chrome Extension, and Samsung SmartThings. Our transferred model can achieve an average F1 score of 90.02% with only handful pieces of labeled data. The transferred model by TKPERM detects 329 different overprivileged applications, such as Tomorrow's Forecast on the way home and YSA email automation from IFTTT, GamingHub and Private Internet Access from Chrome Extension, Button Controller and Hue Mood Lighting from SmartThings.

ACKNOWLEDGMENT

We would like to thank anonymous reviewers for their helpful comments and feedback. This work was supported in part by National Science Foundation (NSF) grants CNS-18-54000, CNS-1920462, CNS-1850479, and OAC-1920462. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF.

REFERENCES

[1] CHRISTOPHER D. MANNING. Dropping common terms: stop words. https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html.

[2] Chrome Extension Crawler. https://robwu.nl.

[3] Chrome Extension Webstore. https://chrome.google.com/webstore/category/extensions?hl=en.

[4] Google Play Store. https://play.google.com/store.

[5] If this then that. https://ifttt.com.

[6] Samsung SmartThings. https://www.smartthings.com/.

[7] ACAR, Y., BACKES, M., BUGIEL, S., FAHL, S., MCDANIEL, P., AND SMITH, M. Sok: Lessons learned from android security research for appified software platforms. In *2016 IEEE Symposium on Security and Privacy (SP)* (2016), IEEE, pp. 433–451.

[8] AFRIDI, M. J., ROSS, A., AND SHAPIRO, E. M. On automated source selection for transfer learning in convolutional neural networks. *Pattern recognition 73* (2018), 65–75.

[9] AU, K. W. Y., ZHOU, Y. F., HUANG, Z., AND LIE, D. Pscout: analyzing the android permission specification. In *Proceedings of the 2012 ACM conference on Computer and communications security* (2012), ACM, pp. 217–228.

[10] BATISTA, G. E., PRATI, R. C., AND MONARD, M. C. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter 6*, 1 (2004), 20–29.

[11] BEN-DAVID, S., BLITZER, J., CRAMMER, K., KULESZA, A., PEREIRA, F., AND VAUGHAN, J. W. A theory of learning from different domains. *Machine learning 79*, 1-2 (2010), 151–175.

[12] BOJANOWSKI, P., GRAVE, E., JOULIN, A., AND MIKOLOV, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics 5* (2017), 135–146.

[13] BOWMAN, S. R., ANGELI, G., POTTS, C., AND MANNING, C. D. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* (2015).

[14] CONNEAU, A., KIELA, D., SCHWENK, H., BARRAULT, L., AND BORDES, A. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (Copenhagen, Denmark, September 2017), Association for Computational Linguistics, pp. 670–680.

[15] DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[16] DONAHUE, J., JIA, Y., VINYALS, O., HOFFMAN, J., ZHANG, N., TZENG, E., AND DARRELL, T. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning* (2014), pp. 647–655.

[17] ENCK, W., GILBERT, P., HAN, S., TENDULKAR, V., CHUN, B.-G., COX, L. P., JUNG, J., MCDANIEL, P., AND SHETH, A. N. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS) 32*, 2 (2014), 5.

[18] FELT, A. P., CHIN, E., HANNA, S., SONG, D., AND WAGNER, D. Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security* (2011), ACM, pp. 627–638.

[19] FELT, A. P., GREENWOOD, K., AND WAGNER, D. The effectiveness of application permissions. In *Proceedings of the 2nd USENIX conference on Web application development* (2011), pp. 7–7.

[20] FERNANDES, E., JUNG, J., AND PRAKASH, A. Security analysis of emerging smart home applications. In *2016 IEEE Symposium on Security and Privacy (SP)* (2016), IEEE, pp. 636–654.

[21] FERNANDES, E., PAUPORE, J., RAHMATI, A., SIMIONATO, D., CONTI, M., AND PRAKASH, A. Flowfence: Practical data protection for emerging iot application frameworks. In *25th USENIX Security Symposium (USENIX Security 16)* (2016), pp. 531–548.

[22] FLEISS, J. L., COHEN, J., AND EVERITT, B. S. Large sample standard errors of kappa and weighted kappa. *Psychological bulletin 72*, 5 (1969), 323.

[23] HOWARD, J., AND RUDER, S. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2018), pp. 328–339.

[24] JIA, Y. J., CHEN, Q. A., WANG, S., RAHMATI, A., FERNANDES, E., MAO, Z. M., PRAKASH, A., AND UNVIERSITY, S. J. Contexlot: Towards providing contextual integrity to appified iot platforms. In *NDSS* (2017).

[25] KANG, Z., GRAUMAN, K., AND SHA, F. Learning with whom to share in multi-task feature learning. In *ICML* (2011), vol. 2, p. 4.

[26] KIROS, R., ZHU, Y., SALAKHUTDINOV, R. R., ZEMEL, R., URTASUN, R., TORRALBA, A., AND FIDLER, S. Skip-thought vectors. In *Advances in neural information processing systems* (2015), pp. 3294–3302.

[27] KULLBACK, S. *Information theory and statistics*. Courier Corporation, 1997.

[28] LIU, F., FELLA, N. L., AND LIAO, K. Modeling language vagueness in privacy policies using deep neural networks. In *Proc. of AAAI'16* (2016).

[29] LONG, M., CAO, Y., WANG, J., AND JORDAN, M. I. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791* (2015).

[30] MURPHEY, Y. L., GUO, H., AND FELDKAMP, L. A. Neural learning from unbalanced data. *Applied Intelligence 21*, 2 (2004), 117–128.

[31] NAN, Y., YANG, Z., YANG, M., ZHOU, S., ZHANG, Y., GU, G., WANG, X., AND SUN, L. Identifying User-Input privacy in mobile applications at a large scale. *IEEE Trans. Inf. Forensics Secur.* (2017).

[32] NAUMAN, M., KHAN, S., AND ZHANG, X. Apex: extending android permission model and enforcement with user-defined runtime constraints. In *Proceedings of the 5th ACM symposium on information, computer and communications security* (2010), ACM, pp. 328–332.

[33] NAVEED, M., ZHOU, X.-Y., DEMETRIOU, S., WANG, X., AND GUNTER, C. A. Inside job: Understanding and mitigating the threat of external device mis-binding on android. In *NDSS* (2014).

[34] OQUAB, M., BOTTOU, L., LAPTEV, I., AND SIVIC, J. Learning and transferring mid-level image representations using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2014).

[35] PAN, S. J., KWOK, J. T., YANG, Q., ET AL. Transfer learning via dimensionality reduction. In *AAAI* (2008), vol. 8, pp. 677–682.

[36] PAN, X., CAO, Y., DU, X., HE, B., FANG, G., SHAO, R., AND CHEN, Y. Flowcog: context-aware semantics extraction and analysis of information flow leaks in android apps. In *Proc. of USENIX Security'18* (2018).

[37] PANDITA, R., XIAO, X., YANG, W., ENCK, W., AND XIE, T. Whyper: Towards automating risk assessment of mobile applications. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)* (2013), pp. 527–542.

[38] PENNINGTON, J., SOCHER, R., AND MANNING, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543.

[39] PETERS, M. E., NEUMANN, M., IYYER, M., GARDNER, M., CLARK, C., LEE, K., AND ZETTLEMOYER, L. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[40] QU, Z., RASTOGI, V., ZHANG, X., CHEN, Y., ZHU, T., AND CHEN, Z. Autocog: Measuring the description-to-permission fidelity in android applications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 1354–1365.

[41] RAHMATI, A., AND MADHYASTHA, H. V. Context-specific access control: Conforming permissions with user expectations. In *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices* (2015), ACM, pp. 75–80.

[42] RAINA, R., BATTLE, A., LEE, H., PACKER, B., AND NG, A. Y. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning* (2007), ACM, pp. 759–766.

[43] ROSENSTEIN, M. T., MARX, Z., KAELBLING, L. P., AND DIETTERICH, T. G. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning* (2005), vol. 898, p. 3.

[44] SADEH, N., ACQUISTI, A., BREAUX, T. D., CRANOR, L. F., MCDONALD, A. M., REIDENBERG, J. R., SMITH, N. A., LIU, F., RUSSELL, N. C., SCHAUB, F., ET AL. The usable privacy policy project. Tech. rep., Technical report, Technical Report, CMU-ISR-13-119, Carnegie Mellon University, 2013.

[45] SAINATH, T. N., VINYALS, O., SENIOR, A., AND SAK, H. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2015), IEEE, pp. 4580–4584.

[46] SHARIF RAZAVIAN, A., AZIZPOUR, H., SULLIVAN, J., AND CARLSSON, S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2014), pp. 806–813.

[47] SUN, B., FUJINO, A., AND MORI, T. Poster: Toward automating the generation of malware analysis reports using the sandbox logs. In *Proc. of CCS'16* (2016).

[48] TAN, J., NGUYEN, K., THEODORIDES, M., NEGRÓN-ARROYO, H., THOMPSON, C., EGELMAN, S., AND WAGNER, D. The effect of developer-specified explanations for permission requests on smartphone user behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), ACM, pp. 91–100.

[49] TIAN, Y., ZHANG, N., LIN, Y.-H., WANG, X., UR, B., GUO, X., AND TAGUE, P. Smartauth: User-centered authorization for the internet of things. In *26th USENIX Security Symposium (USENIX Security 17)* (2017), pp. 361–378.

[50] VIERA, A. J., GARRETT, J. M., ET AL. Understanding interobserver agreement: the kappa statistic. *Fam med 37*, 5 (2005), 360–363.

[51] WANG, J., WANG, Z., ZHANG, D., AND YAN, J. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI* (2017), pp. 2915–2921.

[52] WIJESEKERA, P., BAOKAR, A., HOSSEINI, A., EGELMAN, S., WAGNER, D., AND BEZNOSOV, K. Android permissions remystified: A field study on contextual integrity. In *24th USENIX Security Symposium (USENIX Security 15)* (2015), pp. 499–514.

[53] XIANG, E. W., PAN, S. J., PAN, W., SU, J., AND YANG, Q. Source-selection-free transfer learning. In *Twenty-Second International Joint Conference on Artificial Intelligence* (2011).

[54] XU, R., SAÏDI, H., AND ANDERSON, R. Aurasium: Practical policy enforcement for android applications. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)* (2012), pp. 539–552.

[55] YOSINSKI, J., CLUNE, J., BENGIO, Y., AND LIPSON, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems* (2014), pp. 3320–3328.

[56] ZAMIR, A. R., SAX, A., SHEN, W., GUIBAS, L. J., MALIK, J., AND SAVARESE, S. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 3712–3722.

[57] ZOPH, B., VASUDEVAN, V., SHLENS, J., AND LE, Q. V. Learning transferable architectures for scalable image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018).