Towards Stalkerware Detection with Precise Warnings

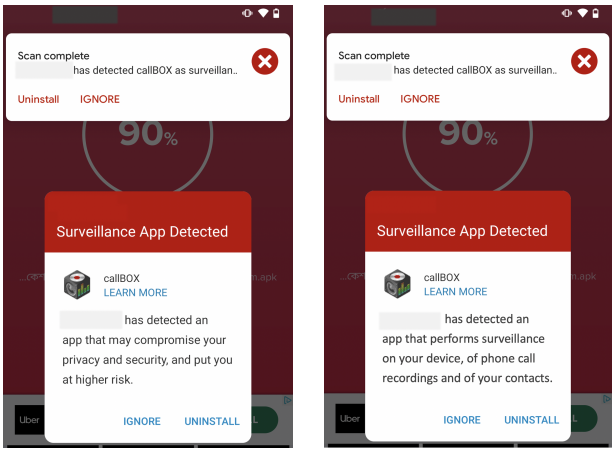
Yufei Han, Kevin A Roundy, Acar Tamersoy

Towards Stalkerware Detection with Precise Warnings

Yufei Han, Kevin A. Roundy, Acar Tamersoy
{Yufei.Han,Kevin.Roundy,Acar.Tamersoy}@nortonlifelock.com
Norton Research Group

ABSTRACT

Stalkerware enables individuals to conduct covert surveillance on a targeted person's device. Android devices are a particularly fertile ground for stalkerware, most of which spy on a single communication channel, sensor, or category of private data, though 27% of stalkerware surveil multiple of private data sources. We present Dosmelt, a system that enables stalkerware warnings that precisely characterize the types of surveillance conducted by Android stalkerware so that surveiled individuals can take appropriate mitigating action. We use an active- and semi-supervised learning to make headway on this task, which is vital because we are the first to characterize stalkerware according to its individual surveillance capabilities at a significant scale, which requires time-consuming expert-labeling of stalkerware apps. Dosmelt leverages the observation that stalkerware differs from other categories of spyware in its open advertising of its surveillance capabilities, which we detect on the basis of the titles and self-descriptions of stalkerware apps that are posted on Android app stores. Dosmelt achieves up to 96% AUC for stalkerware detection with a 91% Macro-F1 score of surveillance capability attribution for stalkerware apps. Dosmelt has detected hundreds of new stalkerware apps that we have added to the Stalkerware Threat List.



(a) Representative warning **(b) Dosmelt's precise warning**

Figure 1: The imprecision of warnings issued by security companies for dual-use apps that can be used for unwanted surveillance may lead to concerns that the precise warnings enabled by our solution Dosmelt would avoid.

1 INTRODUCTION

Intimate Partner Violence (IPV) is a widespread societal problem that results in negative long-term consequences for many of its targets [2, 27, 35]. A 2018 survey conducted in eight Eastern European countries found that 23% of all women experienced physical or sexual IPV and 60% of them experienced psychological IPV [37]. In the U.S., 15.8% of women and 5.3% of men reported being subjected to stalking violence "in which they felt very fearful or believed that they or someone close to them would be harmed or killed" [8]. IPV survivors have shed light on the many ways in which technology plays a role in inter-personal attacks [8, 20, 40, 43, 49], of which tech-enabled stalking and spying by current or former romantic partners are especially common and pernicious [24, 34]. In a recent survey, 10% of the U.S. adult respondents admitted to using a mobile phone app to spy on an intimate partner [50].

While spying apps exist for many kinds of devices, mobile devices are an especially tempting target for attackers, as they accompany individuals nearly everywhere they go, are used for most inter-personal communication, and contain a multitude of sensors. Mobile apps and other software that can be used by an attacker to covertly spy on a targeted individual are typically referred to as *stalkerware* [36]. Studies of stalkerware for mobile phones have identified a variety of mobile spying apps and have manually categorized them into taxonomies that represent different classes of stalkerware apps and distinct stalkerware functionalities [12, 38, 40].

Though these prior studies provided the first set of solutions to this important problem, there are still gaps to fill. First, existing work [12, 38, 40] focuses on the high-level problem of detecting stalkerware, but does not contribute to automatically inferring the individual surveillance capabilities possessed by a stalkerware app. Many vendors of mobile security products that detect stalkerware exhibit similar limitations in that they tend to lump the varied forms of surveillance apps together under a single generic "surveillance", "stalkerware", or "privacy" warning [3]. An example of one such warning is provided in Figure 1a. Generic warnings of this nature are problematic because many of the mobile apps that can be used as stalkerware may have been installed for a legitimate benign purpose. For instance, an app that provides backups of SMS apps and contacts can also be used to spy on a target's text messages. Similarly, an anti-theft app can be used to track a target person's whereabouts. In both cases, the apps are used legitimately when installed on one's own device, but illegitimately when installed covertly on another person's device or on a shared device. The second gap revolves around being limited to a small set of known stalkerware samples. Stalkerware is still a new phenomenon and existing work either performs a focused analysis on a very small number of stalkerware apps [38], or starts off with a small seed set (of IPV-related keywords to search for on app stores or of known stalkerware apps) to identify many candidates in an automated manner, from which false positives are then eliminated through manual validation. The latter is achieved by assuming either a supervised [12] or a graph-based weakly-supervised [40] learning settings. Semi-supervised learning methods like active learning are designed to leverage limited or imprecise sources to provide supervision signal for labeling large amounts of training samples [45], but they are yet to be explored in the stalkerware detection setting.

Taking inspiration from the common phrase, "DOn't Scare ME Like That," we propose DOSMELT, the first stalkerware detection and warning system capable of precisely identifying the individual surveillance methods performed by stalkerware apps. Precise stalkerware notifications are beneficial in two important ways. First, when a stalkerware app is being used for spying by an abusive intimate partner, a precise notification clearly outlines which data has and has not been exposed to the attacker, which is not possible when the security product provides a generic warning such as that of Figure 1a. Precise notifications enable survivors to quickly take appropriate mitigating actions and are less likely to be ignored than a generic warning to which customers can quickly become inured. Second, precise warnings are far less irksome to users who may install an app that monitors sensors or communications for a legitimate, non-abusive purpose. For instance a plumber whose phone routinely sustains water damage may wish to backup SMS messages and contact lists. A precise warning like that of Figure 1b would confirm to the user that the app behaves as expected, whereas the generic warning of Figure 1a raises false alarm bells, leading to unnecessary concern and frustration.

DOSMELT is designed around two important observations about stalkerware. First, stalkerware explicitly advertises its functionality, as it is deliberately intended to collect data from a device for the purpose of making it available for examination, typically by sending it to the cloud or to another device. Second, stalkerware is a diverse term encompassing many kinds of apps. We found that 73% of stalkerware surveils a single sensor (e.g., the microphone or GPS sensor), communication channel (e.g., SMS messages, a social media app's communications), or on-device data (e.g., media, contacts, or browsing history). Warnings that precisely characterize their functionality are more fair to app developers and avoid most of the problems associated with alert fatigue that would arise otherwise.

To provide stalkerware detection that can support nuanced warnings, we formulate a multi-label machine learning problem and build up a labeled dataset using an active learning paradigm to reduce the overhead of stalkerware annotation [55, 58]. Our machine learning task sets each individual surveillance feature as a label to be predicted. Stalkerware apps that implement more than one type of surveillance will have multiple positive labels. This makes the nuanced stalkerware classification task intrinsically a multi-label classification problem [10, 53, 59, 60]. For a practical solution with a small false positive rate, we also classify apps generally as stalkerware vs. non-stalkerware. Our contributions include:

- We present DOSMELT, a multi-label stalkerware detection system that detects stalkerware with far greater nuance than any existing algorithm. Our system supports the creation of precise warnings that enumerate the exact types of surveillance conducted by an app as opposed to imprecise stalkerware warnings that may lead to false alarms unnecessarily for apps with legitimate uses. It demonstrates that an app's self description is sufficient to make headway on this detection task.
- We establish an active learning methodology to promote effective learning with a modest set of hand-annotated Android apps. This is necessary because nuanced stalkerware warning systems do not as yet exist and we are unaware of any large dataset of stalkerware apps for which individual surveillance features had been labeled. Our system integrates a learning-by-prediction strategy that presents human labelers with samples likely to improve the classifier. It then recursively retrains the classifier using the updated training dataset.
- To contribute to the further development of improved stalkerware algorithms, we submitted our manually labeled dataset of stalkerware apps as well as the 246 apps detected by DOSMELT that we also verified to the Coalition Against Stalkerware's [16] Stalkerware Threat List, to which interested researchers and security vendors can gain access with a free membership.

We proceed by providing background information and reviewing related work in Section 2. We then describe our data collection and labeling efforts (Section 3), and present DOSMELT's architecture and its learning strategy in Sections 4 and 5, respectively. Next, we evaluate our solution's ability to detect stalkerware apps and their surveillance features (Section 6). We discuss deployment and opportunities for improvement in Section 7, and conclude in Section 8.

2 BACKGROUND AND RELATED WORK

In building DOSMELT and using it to provide nuanced stalkerware detection, we seek to contribute to a rich multi-disciplinary body of research in intimate partner violence (IPV), spyware and stalkerware detection, and machine learning.

Technology-Enabled IPV. IPV produces damaging long-term effects for its targets, including severe physical violence [37] and homicide [48]. Physical violence is typically accompanied by emotional abuse, which may result in mental health disorders of many kinds [27, 42], such as depression [2], post-traumatic stress disorder [35], low self-esteem [42], and suicidal ideation [19].

Technology-enabled IPV is a troubling phenomenon that fits into the broader ecosystem of online hate, harassment, and abuse [49]. Its manifestations include many forms of harassment [6, 23, 39], character assassination through faked revenge porn [43], impersonation attacks that damage the targeted individual's relationships [25, 40], and above all, spying on an intimate partner through stalkerware and other means, such as knowledge of the survivor's account credentials [24, 25]. The need for stalkerware detection and mitigation strategies is highlighted by a recent survey of the U.S. adults, in which 10% admitted to using an app on a mobile device to spy on an intimate partner [50].

Spyware and Stalkerware. *Spyware* refers to software that collects privacy-sensitive data from its users and the devices on which it is installed [44]. In most cases, the private data collected by spyware goes to corporate entities who monetize it by targeting ads and by selling the data to other entities, though private data may instead be gathered by criminals for identity theft and financial fraud. *Stalkerware*, also known as "intimate partner spyware" [12], is a special case of spyware in which the software collects private data to enable an attacker to secretly spy on a targeted individual and weaponize that data to perpetrate further abuse [12, 24]. Stalkerware also differs from non-stalking spyware in that its extraction of private information from a device is a key feature of the software that it advertises openly. Thus, while non-stalking spyware collects information in secret and silently monetizes it, stalkerware's functionality is only secret and silent when installed on the targeted individual's device, but not in its online advertising or in its

self-description on app stores and other websites. As we shall see, this property of stalkerware was used profitably by Chatterjee et al. [12] in their study of stalkerware.

Despite its clear potential for abuse, the stalkerware phenomenon has largely proven resistant to eradication thus far. One key problem from the legal perspective is that the United States' Federal Wiretapping Law and state laws only make it illegal to sell apps that are "primarily" designed to secretly tap phones, record private conversations or steal emails [15, 30]. As the implications of these laws have become more clear, a definite change came about in stalkerware marketing in the 2018 timeframe, with very few apps since then brazenly advertising under such titles as "Catch My Cheating Girlfriend," and instead touting other use cases while retaining the same functionality [30, 40]. Indeed, many apps that abusers recommend in online forums for use as stalkerware describe themselves as tools for socially acceptable tasks other than spying, such as anti-theft, child online safety, data backups, and the recording of calls, audio, the device's screen, and typed keystrokes [12, 51]. This tactic enables the developers to make claims to plausibly deny that their apps are being used to spy on intimate partners, and thereby to avoid legal responsibility when their apps are used for this purpose. Unfortunately, prominent Android app stores still contain an abundance of location sharing apps, automatic call recording and forwarding apps, SMS backup apps, and even keylogging apps (apps that record all keystrokes) [40].

While we acknowledge that many apps used as stalkerware do also fulfill legitimate use cases, these apps are irresponsible if *they fail to notify all users of the device that their private data is being extracted from the device*. Prominent, repeated notifications differ truly well-intentioned applications from apps with surveillance potential that profit from the stalkerware use case. This key distinction is recognized by the Coalition Against Stalkerware [16] and the Developer Policy for the Google Play app store [28]. Unfortunately, policies about adequate notifications are not well enforced. For example, in June 2021 we reviewed apps on the Google Play app store and found more than 100 "automated" call recording apps advertising the functionality to record all phone calls automatically with no interaction from the device owner. Many of these apps automatically forward call recordings to a pre-specified email address and describe themselves as "hidden."

Spyware and Stalkerware Detection Strategies. Most spyware detection methods rely on dynamic taint tracking of private data, which enables the flow of specific elements of private data to be tracked as it flows through a program until it is exfiltrated over the network [21]. This technique was adopted by commercial software on many platforms where it is used to this day [46], often facilitated by use of the TaintDroid taint tracking tool [22]. Additional spyware detection methods have also emerged, using such methods as static analysis [31] and network traffic analysis [54].

In the realm of stalkerware detection, taint tracking remains useful but insufficient. This is because most spyware starts to silently steal private data in the background as soon as it is installed, whereas stalkerware apps are much more likely to require an initial configuration step as the app needs to know to whom the extracted data should be forwarded. Thus, dynamic analysis is far less likely to succeed on stalkerware, for which automated analysis is unlikely to get past this configuration step.

Chatterjee et al. [12] provided the first published solution designed specifically to detect stalkerware. Their solution issues keyword-based searches like "catch my cheating girlfriend" on the web and app marketplaces to identify candidate stalkerware apps and then uses machine learning to filter out false positives. Taking a different approach, CreepRank [40] uncovered a broad ecosystem of apps used in IPV, which includes stalkerware, but also apps that enable harassment, impersonation, fraud, information theft, and defense against such threats. It constructs a bipartite graph of apps and devices on which they appeared, and it propagates information from a seed set of stalkerware apps to other apps to establish guilt by association. Though both of these algorithms are successful and complementary methods for detecting stalkerware, neither addresses the problem of detecting *how* stalkerware compromises the privacy of an individual to enable the creation of more informative warnings. To accomplish this goal, we build on the taxonomies developed by prior work [38, 40] to create a taxonomy of stalkerware capabilities, and we turn to machine learning methods that can assign an app to multiple categories, each of which represents a distinct stalkerware surveillance capability (e.g., extraction of call logs, web browsing history, and social media messages).

Multi-label Learning and Semi-Supervised Learning. The nuanced attribution of surveillance capabilities to stalkerware apps is a multi-label learning problem. Multi-label classification [7, 52, 61] is of paramount importance in a variety of fields, as witnessed in document classification and automatic tagging of image contents. For example, an image may have *cloud*, *tree* and *sky* tags [7]; one document may contain texts of multiple themes.

A common approach in multi-label learning is to decompose the problem into multiple independent binary classification problems, one for each category. The final set of labels for each instance can then be determined by combining the classification results from all the binary classifiers. This approach is flexible in its ability to use different binary classifiers to build a multi-label learning system. However, it also ignores the underlying mutual correlations among different categories, which can contribute to the classification performance [13, 63]. In our study, we follow the spirit of [47] and use tree structures for this task. Specifically, we use extreme random trees [26] and random forests [9].

Our work also adopts semi-supervised learning. In this learning setting, only a fraction of the training examples are labeled. The goal is to refine the decision boundary using the statistical characteristics of the data distribution conveyed by the unlabelled data instances. Label propagation is a style of semi-supervised learning [56, 62, 64] that has been applied to stalkerware detection [40] in which examples with label information is propagated over a similarity graph built over training data instances. Our study adopts active learning [17, 41, 45], a different branch of semi-supervised learning which interactively queries an external oracle (such as a human annotator) to label new data instances. It is especially suitable for the scenario where unlabeled data is abundant but labeling data manually is expensive. DOSMELT uses a learning-by-prediction based active learning mechanism to incrementally update the nuanced stalkerware classifier. While most variants of active learning choose to submit low-confidence instances to an expert overseer for manual ground-truthing, as in the cases of *uncertainty sampling*

Capability	Description
Browsing-History	Remote access to internet browsing history
Call-Logs	Access to call history
Call-Recordings	Recording of calls in a hidden and automated manner
Camera	Remote viewing of device's camera
Contacts	Remote access to phone's list of contacts
Email	Remote access to emails sent to an app on the device
GPS-Tracking	Tracking the GPS location of the victim
Installed-Apps	Listing of apps installed on the device
Keylogging	Tracking of typing input
Media-Extraction	Remote access to photos, videos, and other media
Microphone	Remote listening to device's microphone
Screen	Recording of the device's screen
SMS	Exports SMS/text messages
Social-Media	Access to social media accounts (including chat and messaging) normally tied to a single device

Table 1: Taxonomy of stalkerware capabilities. Our stalkerware detection approach assigns these capabilities to suspected apps by assuming a multi-label classification setting.

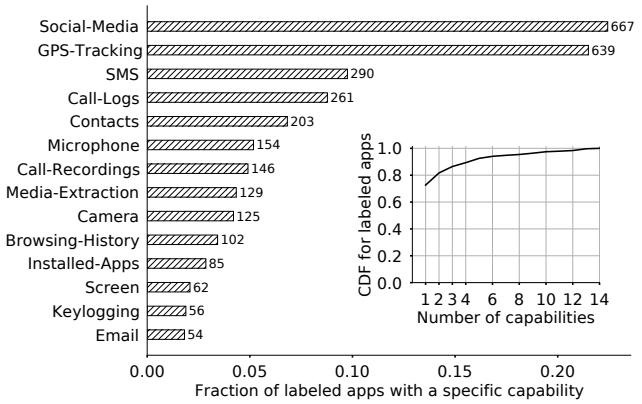


Figure 2: (Outer) Distribution of stalkerware capabilities among the apps manually labeled for this work according to our taxonomy. The numbers next to the bars pertain to the actual counts. Surveillance of social media accounts and locations of victims are the major capabilities. (Inner) Cumulative distribution function of fraction of labeled surveillance apps that have a particular number of capabilities. Most surveillance apps have only one capability.

and *margin sampling*, this process can still leave the human annotators with a significant workload checking questionable samples. Alahmari et al. [1] pioneered an alternative active learning process designed to minimize the annotation workload. It uses an ensemble of highly diversified classifiers that vote, issuing a confidence score that depends on the ensemble's level of agreement. Only the data instances with the highest confidence scores are submitted to human experts for further review. We adopt this active learning mechanism combined with ensemble tree-based classifiers in our framework. The instances our method selects for review help minimize the annotation effort contributed by human experts.

3 DATA COLLECTION AND LABELING

In this section, we discuss our data collection efforts and the process we used to create a labeled dataset of apps categorized according to a taxonomy of stalkerware capabilities.

For this study, we partnered with a large security vendor and obtained two anonymized reports that solely consisted of Android application identifiers that the vendor observed on customer devices between 2019 and 2020. An Android application identifier (app id for short) uniquely identifies an app on a device and in app stores. In line with prior work [12, 40] that also examined Android apps in a similar context, our focus on the Android platform is due to its large market share [5] and to the comparatively large number of stalkerware apps that have been developed for Android devices [29]. Using the vendor's data to indicate apps that were in use on mobile devices, we then queried two app stores with the app ids we possessed, namely APKPure[1] and Google Play Store[2], to retrieve the titles and descriptions of the apps. In total, we obtained information for 1.02 million apps[3]. The app titles and app descriptions are the main pieces of information we rely on to detect stalkerware in this work.

We then developed a taxonomy of stalkerware capabilities to support nuanced detections as follows. We first queried the app titles and app descriptions we had obtained in the previous step using search terms such as "spy" and "track" to identify apps that were likely to be stalkerware. Then, two researchers from our team independently reviewed the app descriptions of a random set of 200 apps identified via this process to examine their capabilities in detail and iteratively refine the taxonomy using inductive coding [33]. We then compared our taxonomy to that of Parsons et al. [38], which also identified different capabilities of a small set of seven overt stalkerware apps, finding that they matched each other well. The final version of our taxonomy of stalkerware capabilities along with their descriptions is shown in Table 1. The main differences between our taxonomy and that of Parsons et al. [38] are that our taxonomy combines surveillance of social media and chat apps into a single category (which we call Social-Media) since our coders observed that a significant majority of the apps advertised the surveillance of both sources. Our taxonomy also omits the calendar-surveillance capability as it was not prevalent in the apps we coded, and we omit the ability to block phone calls as it does not pertain to surveillance.

Once the taxonomy was finalized, the same two researchers participated in a formal coding process consisting of three rounds. In each of the first two rounds, they coded the same set of randomly chosen 150 apps with respect to the taxonomy. Specifically, for each stalkerware capability in the taxonomy, the coders noted whether or not the app under review possessed the capability in a coding sheet, based on an examination of the app title and app description. After each round, the researchers met to discuss any apps they coded differently to better align with each other and improve the coding. In the third and final round, both researchers coded another random set of 100 apps to test agreement. Since our setting is such that each example can be assigned a varying number of codes (each

[1] https://apkpure.com/
[2] https://play.google.com/store/apps
[3] For the majority of the apps, the two app stores contained identical information. When they differed, we retained the longer of the two app descriptions. In the rare case that neither store contained information for a particular app, we attempted to determine if it had a dedicated webpage using search engine queries.

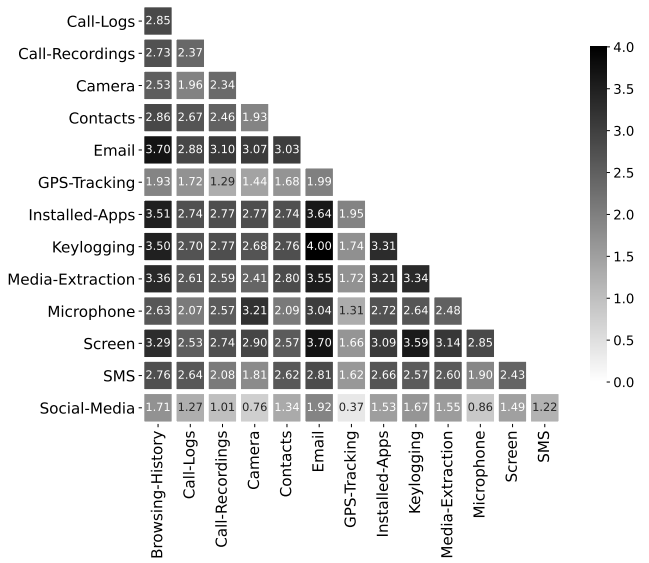


Figure 3: Pointwise mutual information (PMI) values for every pair of surveillance capabilities. PMI is a measure of association, higher values indicate greater association. *Email* is involved in the top-3 strongest associations.

representing a stalkerware capability), we used Krippendorff's alpha, which is a statistical measure of inter-rater reliability suitable for this setting [32]. Our two coders achieved a Krippendorff's alpha of 0.86 in this final round, which indicates strong inter-rater reliability [32]. The team then continued to code stalkerware apps independently. In total, this process led to the labeling of 4,839 apps, where 1,462 of them are stalkerware apps with at least one surveillance capability from the taxonomy of surveillance capabilities. The rest are benign apps.

Figure 2 contains the distribution of stalkerware capabilities among the 1,462 stalkerware apps. Within these apps, the majority support the surveillance of social media accounts (*Social-Media*) and locations of victims (*GPS-Tracking*). Also shown in Figure 2 is the cumulative distribution function of fraction of labeled surveillance apps that have a particular number of capabilities. While 73% of stalkerware apps support only one surveillance capability, there are many apps with multiple capabilities. To examine the associations between surveillance capabilities, we computed the pointwise mutual information (PMI) values for every pair of capabilities, which is defined as follows. Let X and Y be discrete random variables. Then, for a pair of outcomes $x \in X$ and $y \in Y$, PMI(x, y) measures the discrepancy between the probability of their coincidence given their joint distribution and marginal distributions under the assumption of independence, and is defined as PMI(x, y) = $\log \frac{p(x,y)}{p(x)p(y)}$. Higher PMI values indicate greater association between x and y. Figure 3 shows the PMI values for every pair of surveillance capabilities, as captured by our dataset of 4,839 labeled apps. The top-3 strongest associations involve *Email*, and they are between *Email* and the surveillance capabilities of *Keylogging*, *Screen*, and *Browsing-History*, respectively.

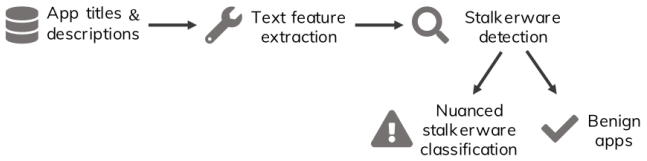


Figure 4: Steps taken by DOSMELT to perform nuanced classification of stalkerware.

4 SYSTEM ARCHITECTURE

The architecture of DOSMELT is shown as a pipeline in Figure 4. As described in Section 3, the input dataset is comprised of the titles and descriptions of the apps. These text-based descriptors are then fed into a feature extraction module. Here, we remove the stop words and build a bag-of-words (BoW) model where each word becomes a token. The *TF-IDF* (term frequency-inverse document frequency) value of each token is then computed, and for a particular app, the combination of these values form the feature vector of the app. Then, a stalkerware detector is applied over these feature vectors to decide whether the app is stalkerware or benign. In the final step, the nuanced classification module identifies the surveillance capabilities carried out by the detected stalkerware apps.

For both the detection and nuanced classification modules, we adopt a learning-by-prediction based active learning paradigm to guide the training process. This is because we are in a setting where the number of labeled examples that can be used during training is minuscule and obtaining new labeled examples is costly, as it requires a human annotator to examine a blob of text about an app. Active learning is suitable for this setting, it requires the labeling of a small fraction of the training dataset in the beginning [45]. After building an initial classifier, active learning helps DOSMELT to iteratively select informative training data instances and retrieve their labels from human experts. These selected data instances enrich the supervision information and gradually refine the classifier's parameters. Alternative to labeling the whole training set, the active learning scheme defines a low-overhead yet effective training process in DOSMELT.

5 FEATURE ENCODING & ACTIVE LEARNING

In this section, we describe how the learning-by-prediction strategy enables us to gradually build an accurate stalkerware classifier with nuanced detections by starting with a small set of labelled apps, each of which is represented with features extracted from their title and description information.

5.1 Preprocessing and Feature Engineering

We propose the use of *Term Frequence-Inverse Document Frequency* (TF-IDF) features derived from the keywords extracted from textual app descriptions. We first use the NLTK (cite) package to remove stop words from the extracted text-based descriptions, as these commonly used stop words (such as "the", "a", "an", "in") contain no information related to stalkerware functionality. After that, we count the occurrence frequency of the remaining key words and exclude those that appear only once in the training data set. These

extremely unusual words can not generate statistically stable TF-IDF features, and they increase the classifier's risk of overfitting to the training data.

Next, we construct a Bag-of-Word model of the words extracted from the training texts and derive the TF-IDF based feature vectors for each training text instance as follows:

$$\text{tfidf}(w,d) = TF(w,d)\log(N/(df+1)) \tag{1}$$

where w denote each word considered in the feature extraction. d represents a document, in our case, a mobile app's self-description. N denotes the number of the apps in the training set. df denotes the document frequency. This is defined as the occurrence frequency of w in the training text instances. $TF(w,d)$ denotes the term frequency, which is defined as the instance count of the word w in the training text document d divided by the number of words contained in d. In the computation of term frequency, all words are considered equally important, though clearly, certain words may appear many times and yet have little importance. Hence the inclusion of the Inverse-Document Frequency (IDF), the second half of the equation, which assigns low weight to frequent terms while scaling up the importance of rare terms.

5.2 Detection with Learning by Prediction

There are three important advantages using active learning in our work. Most fundamentally, no labeled dataset of stalkerware surveillance capabilities is in existence, and constructing one is time consuming. It is vital that human effort is spent where it can be most useful. Second, stalkerware apps are sufficiently rare that randomly sampling Android apps would be unlikely to turn up any meaningful number of stalkerware apps to label. Indeed, the Coalition Against Stalkerware's Stalkerware Threat List contains fewer than ten thousand Android app id's, and security vendors report seeing 10 million or more app id's installed on their customer's devices in a year [40]. Third and finally, active learning not just allows us to iteratively update our classifier's parameters as we go along, but help to adjust our threat detection pipeline by including additional data cleaning and/or improving feature engineering. Our own experience provides examples of why this helps. After the first iterations of active learning with DOSMELT, we made two realizations. First, among the highest ranking applications were many apps in foreign languages that were not being suitably categorized by DOSMELT because of overfitting due to insufficient training data for languages other than English. This induced us to fix our pipeline to improve its foreign-language filtering, limiting our classifier to English for the present time. Second, as a result of these poor outcomes, we came to the determination that we should exclude keywords as features if they appear only once. These two changes dramatically improved the classifier in subsequent iterations of active learning.

The flowchart of DOSMELT's active learning method is illustrated in Algorithm 1. In each iteration, we assume the trained classifier f as an ensemble of base decision models f_k ($k = 1, 2, 3, ..., L$). Each base model f_k is trained to increase the diversity between each other, e.g. by randomly sampling feature subsets in training in Random Forest. f then parses the set of the unlabelled textual app descriptions. For each unlabeled instance, each f_k produces a confidence score normalized between 0 and 1 via the *sigmoid* function. For detection, higher confidence score denotes that it is more likely that the app is stalkerware. For classification, the magnitude of the score measures the decision confidence of tagging an app with a specific surveillance capability. The resultant confidence scores are ranked in a descending order. The unlabelled instances with the top K-ranked confidence scores are then investigated by human analysts. After confirming/correcting the labels of these instances, they are added to the training data set. Finally the classifier f is retrained with the updated training set. The number K of selected instances for manual verification decides the sampling coverage of the active learning method.

Let us denote the textual feature of an unlabelled app as x_i, ($i = 1, 2, 3, ..., |\mathcal{U}|$). $y_i = +1$ if the app is stalkerware (for detection) or has a specific stalkerware capability, and vice versa. The ensemble vote score produced on the unlabelled textual instance can be represented as $S(x_i) = \sum_{k=1}^{M} f_k(x_i)$. The learning-by-prediction method [1] augments the training data set by hand-labeling the unlabelled data instances for which the current detection/classifier outputs the highest $S(x_i)$. Supposing the true class label of x_i as y_i, the classification margin of x_i with respect to the current detection/classifier model f gives as $m_f(x_i, y_i) = y_i(2 * S(x_i) - 1)$. The data instance selection criterion applied in [1] has a two-fold goal. *For the correctly detected/classified instances*, the human annotator confirms the prediction output from the ensemble model. The corresponding instances usually contain very indicative keywords/terms denoting the suspicious surveillance capabilities, which clearly differentiate stalkerware and non-stalkerwre apps. *For the miss-classified instances*, where $m_f(x_i, y_i)$ is negative yet with a large magnitude, the human annotator identifies the miss-classification error and provides their true labels. In this sense, the learning-by-prediction step in [1] converges to the well known principle of *Misclassification Loss Reduction* in active learning [45]:

$$x^* = \arg\max_{x_i} P_\theta(\hat{y}_i = 1|x_i)\ell_f(x_i, y_i) \tag{2}$$

where $\hat{y}_i$ is the predicted label by f. $P_\theta(\hat{y}_i = 1|x)$ denotes the probabilistic decision confidence of the classifier f. $\ell_f(x_i, y_i)$ is the miss-classification loss of the current f over the data instance (x_i, y_i), which is a monotonically decreasing function of the classification margin. For the nuanced classification, since f produces multiple outputs simultaneously, one per surveillance capability, Eq.2 can be instantiated to the multi-label learning scenario:

$$x^* = \arg\max_{x_i} \sum_{j=1}^{m} P_\theta(\hat{y}_{i,j} = +1|x_i)\ell_f(x_i, y_{i,j}) \tag{3}$$

where m denotes the number of the surveillance capabilities involved in the nuanced classification. $\hat{y}_{i,j}$ and $y_{i,j}$ are the predicted label of x_i with respect to the surveillance capability j ($j = 1, 2, 3, ..., m$). The inconsistency between the predicted class labels and the ground truth labels denotes incapability of the current model in capturing the underlying decision boundaries. Adding the these instances for retraining can thus help correct the bias in f causing large miss-classification loss. In our work, the data set we collect to train the stalkerware detectors is heavily skewed towards benign apps due to the rare existence of stalkerware apps. Therefore, the learn-by-prediction process for building the stalkerware detector examines the apps that are confidently classified to be malicious. In the first iterations of the training process, we can find many false alarms

Algorithm 1: Learning-by-prediction Active Learning Framework

Input: Initial training data set $\mathcal{D}$, Untagged textual training instances $\mathcal{M}$, a training paradigm $\mathcal{A}$ for the ensemble classifier, the training rounds m

Output: Learned stalkerware detector/classifier $\mathbf{f}$

$\mathcal{D}^0 \leftarrow \mathcal{D}$;

$\mathcal{M}^0 \leftarrow \mathcal{M}$;

for $t = 1$ **to** m **do**

$\{\mathbf{f}_1^t, \mathbf{f}_2^t, ..., \mathbf{f}_L^t\} \leftarrow \mathcal{A}(\mathcal{D}^{t-1})$;

Generate the voted confidence scores $\mathrm{y} = \{y_i\}(i = 1, 2, 3, ..., |\mathcal{M}^{t-1}|) \leftarrow \sum_{i=1}^{L} \mathbf{f}_i^t(\mathcal{M}^{t-1})$;

Select the top K text instances (denoted as S^t) with the largest y_i from $\mathcal{M}$;

Let human analysts verify the true labels of the text samples in S;

Update the training text sample set $\mathcal{D}$: $\mathcal{D}^t \leftarrow \mathcal{D}^{t-1} + S^t$, with the manually verified labels for S;

$\mathcal{M}^t \leftarrow \mathcal{M}^{t-1} - S^t$;

end

return the ensemble classifier $\{\mathbf{f}_1^m, \mathbf{f}_2^m, ..., \mathbf{f}_L^m\}$;

in these top-ranked apps. Correcting the miss-classification errors then helps the detector to refine the boundary in the textual feature space between the stalkerware and benign apps.

6 EXPERIMENTAL EVALUATION

We evaluate the performance of our method with a dataset composed of textual features for Android apps (see Section 3). Next, we describe our experimental setup, cover the stalkerware detection results, review the nuanced stalkerware capability detection results, and provide insights into the classifier.

6.1 Experimental Setup

Our experimental setup consists of two components.

- **Stalkerware detection**: Detecting whether an app is an instance of stalkerware is a classic binary classification task that we approach with active learning because of the high cost of labeling data for this purpose. When our detection module produces a positive label, it indicates that an input app contains stalkerware functionality.
- **Stalkerware capability detection**: We further identify the surveillance capabilities possessed by each stalkerware app based on its textual features. In this classification task, the presence of a specific surveillance capability is indicated by a positive label. Note that some stalkerware apps might implement multiple surveillance capabilities. This nuanced classification task is thus an instance of multi-label classification, i.e., each data instance can carry multiple labels.

For the stalkerware detection task, we experiment both with Random Forests (*RF*) of 500 trees and Gradient Boosted Trees (*GBDT*) with 150 cascade layers as our ensemble detection model. Empirically, this setting provides a stable detection accuracy. For the stalkerware capability detection task, we use *RF* and Extremely Randomized Trees (*Extra-Tree*) to build the ensemble classifier. Similar to *RF*, *Extra-Tree* picks a random subset of candidate features for each tree. Instead of looking for the most discriminative thresholds for a tree split, the thresholds are drawn at random for each candidate feature and the best of them is used as the splitting rule. This allows to reduce the variance of the model, at the expense of a slightly greater increase in bias.

In general, the tree-ensemble based methods are easy-to-tune and generalize well across many data mining scenarios [9]. Furthermore, *RF* and *GBDT* conduct model training and ranking of the feature importance via measuring out-of-bag error in parallel. The feature importance evaluation returned by the tree ensembles reflects the informativeness of each keyword in the textual feature space. While deep neural architectures have a potential to improve classification performance in our setting, we leave this endeavour as future work.

	Accuracy		AUC		
Round	**RF**	**GBDT**	**RF**	**GBDT**	**Number of labeled instances**
0	0.763	0.735	0.760	0.740	200/400 (stalkerware/benign apps)
1	0.900	0.880	0.897	0.865	400/470 (stalkerware/benign apps)
2	0.960	0.940	0.960	0.932	600/520 (stalkerware/benign apps)
Baseline	0.975	0.960	0.970	0.942	All labeled instances

Table 2: Stalkerware detection results of different training rounds in DOSMELT using Random Forest (RF) and Gradient Boosted Trees (GBDT).

6.2 Stalkerware Detection with Active Learning

We conduct two sets of studies to evaluate the overall performance of the DOSMELT pipeline at the stalkerware prediction task and its ability to learn quickly through active learning. First we present a careful measurement of our active learning methodology using a cross-validation experiments. Second, we present a more practical application of our active learning methodology in which we use active learning to improve DOSMELT's ability to generalize to stalkerware detection beyond our carefully curated dataset of labeled stalkerware apps.

A cross-validation study of DOSMELT's stalkerware detection using active learning. To measure DOSMELT's stalkerware detection accuracy, we set aside 30% of the labeled dataset of Android apps as an independent testing set and use the remaining 70% of the labeled data for training. We repeat this training-testing split 5 times. The derived detection and classification performance metrics are averaged and reported in Table 2. In particular, Table 2 shows the result of a series of experiments we conducted to evaluate DOSMELT's active learning methodology. Each of the three rounds of active learning use increasing amounts of labeled data during training. In the first iteration we allow the classifier access to only a small amount of labeled stalkerware and non-stalkerware apps, with the remaining training samples are hidden from the classifier and are considered as unlabelled instances. At the end of each round, DOSMELT selects the unlabelled instances with confidence score larger than 0.9 as the next apps to be labeled by human analysts, which are added to the training set in the subsequent round. Thus, we expect to see improved classification accuracy in subsequent rounds as the number of labeled apps increases.

In Table 2, the detection performance is indicated by *ACC* and *AUC*, which denote the accuracy score and the AUC-ROC score of the detection model, respectively. "Baseline" is the performance derived using all the training instances. We introduce this baseline as a reference to verify the effectiveness of the learning-by-prediction

Round	Stalkerware	Total	Accuracy
1	24	198	11%
2	124	200	62%
3	48	116	41%

Table 3: Accuracy results for 3 rounds of active learning conducted to improve DosMelt's ability to generalize. We applied DosMelt to 1.56 million unlabeled apps and hand-labeled its most confident detections in each round. Accuracy improved until the third round, when DosMelt seemed to run out of stalkerware apps to detect.

	Macro-F1		Micro-F1		
Round	RF	Extra-Tree	RF	Extra-Tree	Number of labeled instances
0	0.732	0.735	0.716	0.723	409
1	0.840	0.838	0.837	0.842	615
2	0.914	0.910	0.910	0.910	830
Baseline	0.925	0.925	0.907	0.905	All labeled instances

Table 4: Multi-label stalkerware capability classification results across different training rounds using Random Forest (RF) and Extremely Randomized Trees (Extra-Tree).

active learning technique. The initial detection *ACC* and *AUC* (training round 0) is relatively low (lower than 0.8) due to the limited training data set. The detection accuracy increases consistently and significantly in the training rounds 1 and 2, producing *ACC* and *AUC* scores higher than 0.95, which confirms the merits of the active learning method. DosMelt avoids exhaustive labeling efforts in favor of a focus on the most informative data instances to correct the bias in the estimate of the decision boundary. It thus minimizes the overhead of labelling the training samples. As seen in Table 2, when only 60% of the training instances are used to build the detection model, the detection accuracy is already comparable to that of the baseline, which is trained on the full training set.

In subsequent training rounds (rounds 1 and 2), 200 stalkerware apps are selected from the unlabelled instances for retraining via the learning-by-prediction method. In contrast, 70 and 50 non-stalkerware apps are selected in rounds 1 and 2, respectively. There are some false alarms with a decision confidence score larger than 0.9. A possible explanation for these false alarms is that recognizing the stalkerware with the *TF-IDF* based features depends on the occurrence of the keywords or phrases relevant with the surveillance functions. For example, "keystroke" is indicative of potential surveillance use in the *RF* and *GBDT* based detection models. This word is usually used to introduce the keystroke logging function in keylogger apps for stalking use. The occurrence of the word tends to increase the decision confidence of classifying the corresponding app as stalkerware. However, this word can also be found in the descriptive texts of benign applications, and with limited labelled training samples, the detection model is prone to overfit and to overestimate the importance of sensitive words. It is also prone to incorrectly attach importance to any word that happens by chance to appear in the description of a stalkerware app but not elsewhere even if this word does not relate to surveillance. This issue arises despite our practice of not assigning *TF-IDF* weights to words that occur only once in our dataset, but fortunately it becomes less common as more training data added to the model through active learning. In Section 6.4, we will discuss the keywords that are considered by the *RF*-based detector as the most informative features for stalkerware detection.

Improving DosMelt's ability to generalize. Once we had managed to achieved good classification results with DosMelt on our labeled data set, we again turned to active learning to improve its ability to generalize to as-yet undetected Android stalkerware. To this end, we trained a Random Forest classifier on our full labeled dataset of Android apps and performed three rounds of a learning-by-prediction exercise against the rest of the many apps that were not labeled, but for which we had app title and description information (see Section 3). At the time of this experiment, DosMelt's accuracy in the binary stalkerware classification task was already above 97%. Even so, its generalization ability left much to be desired. As seen in Table 3, we achieved much improved results in the model's ability to detect undiscovered stalkerware over three iterations of active learning, thanks both to the improved labels, and to an obvious need for improvements to the DosMelt pipeline that became apparent after the first round of active learning. In the first round of hand-labeling apps identified through active-learning, we noticed that the broad set of 1.56 million Android apps contained many benign foreign language apps that DosMelt was mistakenly classifying as stalkerware. This was particularly problematic because the training dataset consisted of hand-coded apps from which foreign-language apps had generally been excluded, which resulted in the model attaching high importance to certain foreign-language words that have nothing to do with stalkerware. Accordingly, we added a foreign-language filter, which dramatically improved DosMelt's ability to generalize in the second round. In the third round, DosMelt seemed to run out of stalkerware apps to classify after a certain point. It correctly identified 19 stalkerware apps among its top 20 most confident predictions, but only identified 3 stalkerware apps among the 20 least confident predictions that we hand-labeled. Though DosMelt seemed to have approached a limit in its ability to identify new stalkerware apps, additional runs are constantly needed to detect new stalkerware apps, because of the high turnover rate among such apps [40]. When we uploaded all of these manually verified detections to the Coalition Against Stalkerware's Stalkerware Threat List, we found less than a 3% overlap with the coalition's set of previously detected apps, which suggests that mining app titles and descriptions to detect stalkerware as DosMelt is a strong complement to existing commercialized methods for detecting stalkerware.

6.3 Nuanced Stalkerware Capability Classification

We now turn to the novel and challenging task of identifying the surveillance capabilities of stalkerware apps using limited amounts of labeled data in an active learning framework. For this classification task, we use 438 stalkerware apps (30% of the stalkerware apps) as our testing set, leaving 1,024 labeled stalkerware apps for training. However, as before, we evaluate our active learning capabilities by initially training with the description texts of only 40% of these labeled apps (i.e., 409 stalkerware apps). Note that we use more labelled training instances at the initial step of the training process for the multi-label nuanced classification task as compared to the binary stalkerware detection task because we wish

Capability	ACC	AUC	Positive label fraction
Browsing-History	0.905	0.882	0.019
Call-Logs	0.967	0.937	0.051
Call-Recordings	0.890	0.877	0.027
Camera	0.961	0.935	0.022
Contacts	0.882	0.850	0.039
Email	0.909	0.879	0.009
GPS-Tracking	0.971	0.937	0.099
Installed-Apps	0.857	0.820	0.016
Keylogging	0.968	0.940	0.009
Media-Extraction	0.981	0.942	0.024
Microphone	0.934	0.907	0.030
Screen	0.966	0.935	0.012
SMS	0.988	0.960	0.055
Social-Media	0.983	0.942	0.124

Table 5: Nuanced stalkerware classification results for each surveillance capability using Random Forests *RF*.

	Counts			Counts	
Keyword	Stalk.	Non-stalk.	Keyword	Stalk.	Non-stalk.
keymonitor	12	0	intercept	5	1
whatweb	11	0	transcript	5	1
biometrics	7	0	memos	2	1
calendar	30	8	infrared	2	1
clone	20	3	keystrokes	2	1
database restore	6	0	hider	2	1
whatsmessage	6	0	dialing	12	5
phonefind	6	0	chatting	24	13
spy	6	0	gps	128	70
espiar	5	0	message	207	157
read_phone_state	5	0	locate	39	37
stealer	3	0	email	6	4
monitored display	2	0			

Table 6: Top 25 ranked keywords for stalkerware detection derived using RF.

to ensure that each of the 14 surveillance functions appears at least three times in the labelled training instances. In each round of the learning-by-prediction process, more stalkerware apps with their surveillance capabilities are confirmed by human annotators are added to update the surveillance capability classifier.

We report the results of active learning in Table 4, however this time we use *Macro-F1* and *Micro-F1* scores to measure the accuracy of our multi-label classifier on the testing instances. Again, we divide the dataset into testing and training data 5 times at random and report the averaged results. The table shows that classification effectiveness consistently increases with each training round as more training instances are added. The accuracy stabilizes once 52% of the training instances are selected for model training, regardless of the classifier's architecture.

Compared to the results of binary stalkerware detection in Table 2, surveillance-capability classification requires more labelled training instances at the initial training step due to the nature of multi-label learning, in which success depends on the ability to capture the correlation between different labels [10, 53, 59, 60]. In our case, the classifier must learn the relationships and co-occurrences of different surveillance capabilities in the training set. Therefore, it requires more labeled training instances so that it can identify the statistical correlations between surveillance capability labels.

In Table 5, we also show the *ACC* and *AUC* scores per surveillance capability achieved after the second round of active learning. For each surveillance capability label, we also give the fraction of stalkerware in our dataset with this surveillance capability, provided under the *Positive label fraction* column. A lower value denotes that the corresponding surveillance capability appears less.

As shown in Table 5, the sparsity level of the surveillance label is associated with the classification accuracy with respect to the corresponding surveillance function. The surveillance labels with the positive label ratio higher than 0.02 in general have *ACC* and *AUC* scores higher than 0.93. With more label occurrences, the classifier can capture more stable correlations between the *TF-IDF* text features and the labels it attempts to predict. The two accuracy metric values of *Call-Recordings* and *Contacts* are exceptionally low, under 0.9. One likely explanation for this observation is that the key words describing the function of recording incoming calls and remote access to the contacts (like "dialing", "memo" and "chatting") can be also found in the text descriptions of apps attributed to the other surveillance types, such as *Screen* and *Social-Media*. These keywords do not provide enough confidence to enable accurate detection of these surveillance capabilities. Inversely, the *ACC* score of *Keylogging* is higher than 0.96. Texts of the stalkerware apps with keystroke logging functions contain the indicative key words such as "keymonitor" and/or "keystrokes". These keywords are exclusively observed in the *Keylogging* type. It is thus easy to attribute this surveillance type accurately.

6.4 Importance Ranking of the Keywords

To deepen our understanding about the nuanced classification mechanism, we evaluate the informativeness of each textual feature dimension using random forest with the criterion of *mean decrease impurity*. In our study, each feature adopted in the *RF*-based detector is the *TF-IDF* value of a key word extracted from the description texts of the apps. We rank the key words according to feature informativeness and provide the top 25 ranked key words in Table 6. Furthermore, we count the occurrence frequency of each key word in the stalkerware and non-stalkerware apps of the collected data set. These frequencies are noted as *Counts in stalkerware apps* and *Counts in non-stalkerware apps* in Table 6. We record the two counts to illustrate how the top-ranked key words trigger detection of stalkerware apps.

In general, the top-ranked keywords are more indicative of the text-based descriptions of stalkerware than of non-stalkerware apps. For example, the following terms only exist in the titles and descriptions of stalkerware apps: `keymonitor`, `whatweb`, `clone`,`biometrics`, `database restore`, `whatsmessage`, `phonefind`, `spyhuman`, `espiar`, and `stealer`. The key word "keymonitor" is the name of a popular stalkerware appears that appears many times in the app stores under different app identifiers *ikeymonitor* [4]. It stands out because as one of the most fully-featured stalkerware apps we have observed, covering messages from Facebook Messenger, Line, Hangouts, Viber, WeChat, Kik and Skype, if these applications are installed. It also enables access to call history logs, call recordings, calendar events, screen captures, audio recordings, contact lists, GPS locations, online browsing history, notes, pasteboard data and keystrokes. Another unsurprisingly prominent keyword is `spy`, which appears very often in the title and descriptions of stalkerware, and even

as a substring in app titles such as *mSpy* and *spyhuman*. These two apps conduct a broad spectrum of surveillance functions including copying call logs, tracking the app installation history and copying text messages. These stalkerware apps provide good examples showing the multi-label nature of the nuanced stalkerware classification: *a stalkerware app is usually equipped with multiple surveillance activities at the same time.* In contrast, other prominent keywords that correspond to app titles are for apps that spy on a single communication channel, such as the WhatsApp social-media app: `whatsmessage`, `whatweb`, and *whatweb cloner*. Each of these apps enable an attacker to spy on an individual's WhatsApp account on another device. A few other noteworthy examples are the key words *clone*, *biometrics*, *phonefind* and *stealer*, each of which indicate a surveillance modality or style of privacy theft. It is interesting that some foreign language keywords did appear in the list, including the key word `espiar`, which is the Spanish translation of "spy." Finally, there are few keywords in the list that appear both in stalkerwrae and non-stalkerware apps, such as `dialing`, `chatting`, `gps`, `message`, `locate` and `email`. Indeed, these keywords are associated with the essential services of mobile devices. They can be potentially used in the description texts of both benign and malicious apps. Combined with other indicative key words, they can then form a description of the surveillance behavior. In the appendix, we give 19 concrete examples of detected stalkerware apps by Dosmelt in Figure.5. We also show the identified surveillance capabilities of these stalkerware apps, which are verified manually by human experts. The surveillance attribution results demonstrate the detailed surveillance warnings produced by Dosmelt

7 DISCUSSION

We provide a brief discussion around the deployment plans of Dosmelt, its limitations and a few pointers for future work.

Impact. Our experimental evaluation suggest that Dosmelt's mining of app titles and descriptions is a useful means of detecting stalkerware apps and identifying their surveillance capabilities. Though this method would be insufficient as a standalone stalkerware detection method, its ability to detect quickly detect new apps based on their advertised functionality seems a good complement to existing static and dynamic analysis techniques [21, 31], and to reputation systems [40] for detecting stalkerware, with the crucial benefit of being the first system to detect individual stalkerware surveillance capabilities.

To help the survivors of IPV, we submitted the manually labeled dataset we constructed, which consists of several hundred stalkerware apps that did not previously exist in the the Coalition Against Stalkerware's [16] Stalkerware Threat List, which includes 246 apps directly detected by Dosmelt. The Coalition's threat list is used as a source of detections by several security vendors to protect their customers from stalkerware.

Dosmelt is notable because it enables the creation of precise stalkerware warnings based on app capabilities, as shown in Figure 1. While we believe that this represents a substantial leap forward, it only solves one of several challenges in providing well-considered, usable stalkerware warnings. Consider for instance, that for a warning to be useful in intimate partner violence settings, we must consider that it is the abuser who will be installing the stalkerware app on the targeted individual's device [24]. Thus, an immediate warning by an installed security app is likely to be counter-productive, the goal being to warn the device's owner, and all other users of the device *other than* the person who installed the app, as this individual is already aware of the app's functionality and would be likely to uninstall the security application. Furthermore, the language used in the app itself should be carefully considered, as it is designed to be seen by IPV survivors, who may be suffering from Post-Traumatic Stress [35] and other mental health complications as a result of ongoing abuse [27]. As important as these notification-related challenges are, we leave them out of scope for now, and plan to address them in our future work.

Limitations. We designed Dosmelt to test the limits of an app's self-description as a way to detect new stalkerware apps. While this method has clear advantages, it also comes with obvious limitations. First, this method works best for apps that have at least at one point been hosted on the Google Play app store. While we are able to extract descriptions from stalkerware that hosts its self-descriptions on its own website, the extraction of these descriptions for newly detected stalkerware would be very difficult to automate. Additional, complementary methods of stalkerware detection such as dynamic analysis [21] and reputation-based systems [40] are needed to assist in this case. Dynamic analysis methods could be used to identify some of an app's stalkerware capabilities, but these methods do not distinguish between stalkerware and non-stalking spyware.

A second obvious limitation of our approach is the possibility of adversarial attacks. The most obvious attack to conduct against Dosmelt or a similar approach based on an app's self-description, would be to avoid stalkerware-related keywords, though this approach is disadvantageous to the developer because those same keywords enable abusers to discover the app through internet and app store searches. Alternatively, a stalkerware developer might gain access to a list of keywords that Dosmelt associates with benign apps, and inject benign keywords into the title or description of the app to weaken the classifier's confidence in its prediction. While this is certainly a feasible attack, a more adversarially-resistant version of Dosmelt could perhaps be based on recent work in adversarially robust decision trees [11, 14] to form a resilient random forest model. We intend to explore these directions in future work, though ultimately, we recommend Dosmelt as a complement to existing techniques like dynamic tracking of sensitive information flow within the app [18, 22, 57] and not as a standalone solution.

A third clear limitation of our work is that we evaluated Dosmelt only on apps with titles and description in English (because the training data that we had access to was English-centric). There is a clear need for extending the stalkerware detection methods like ours to be applied to non-English apps. We will address this gap in our future work.

8 CONCLUSION

Stalkerware is a global phenomenon, having both security and privacy implications for an increasing number of individuals. Existing work has not attempted to automate the identification of various surveillance capabilities of stalkerware apps, which results in confusing stalkerware notifications, particularly for apps that can be used as stalkerware but have other benign use cases. It is challenging to perform this task at scale however, as there are no large datasets of stalkerware apps for which individual surveillance

features had been labeled. To this end, we developed DOSMELT, a multi-label stalkerware detection system that detects stalkerware with in a nuanced way, by enumerating the types of surveillance that the app conducts. Our method leverages an active learning methodology to promote effective learning with a modest set of hand-annotated apps. We conducted an extensive evaluation of DOSMELT showing its effectiveness, and we submitted our manually labeled dataset of stalkerware apps and those detected by DOSMELT to the Stalkerware Threat List, to which interested researchers can apply for access.

REFERENCES

[1] Saeed Alahmari, Dmitry Goldgof, Lawrence Hall, Palak Dave, Hady Ahmady Phoulady, and Peter Mouton. 2018. Iterative Deep Learning Based Unbiased Stereology with Human-in-the-Loop. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 665–670. https://doi.org/10.1109/ICMLA.2018.00106

[2] Deborah K Anderson, Daniel G Saunders, Mieko Yoshihama, Deborah I Bybee, and Cris M Sullivan. 2003. Long-term trends in depression among women separated from abusive partners. *Violence against women* 9, 7 (2003), 807–838.

[3] AV Comparatives. 2020. Stalkerware Test 2020. https://www.av-comparatives.org/reports/android-stalkerware-report-2021/.

[4] Avira. 2020. Avira is on the hunt for Stalkerware. https://www.avira.com/en/blog/on-the-hunt-for-stalkerware.

[5] Jason Bays and Umit Karabiyik. 2019. Forensic analysis of third party location applications in android and ios. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 1–6.

[6] Lindsay Blackwell, Jill Dimond, Sarita Schoenebeck, and Cliff Lampe. 2017. Classification and Its Consequences for Online Harassment: Design Insights from Heartmob. *Proceedings of the ACM on Human-Computer Interaction*, Article 24 (2017). https://doi.org/10.1145/3134659

[7] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition* 37, 9 (2004), 1757–1771. https://doi.org/10.1016/j.patcog.2004.03.009

[8] Matthew J. Breiding, J. Chen, and M. C. Black. 2014. *Intimate Partner Violence in the United States — 2010.* Technical Report. Atlanta, GA: National Center for Injury Prevention and Control, Centers for Disease Control and Prevention.

[9] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[10] Serhat Selcuk Bucak, Rong Jin, and Anil K Jain. 2011. Multi-label learning with incomplete class assignments. In *CVPR*. 2801–2808. https://doi.org/10.1109/CVPR.2011.5995734

[11] Stefano Calzavara, Claudio Lucchese, and Gabriele Tolomei. 2019. Adversarial training of gradient-boosted decision trees. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2429–2432.

[12] Rahul Chatterjee, Periwinkle Doerfler, Hadas Orgad, Sam Havron, Jackeline Palmer, Diana Freed, Karen Levy, Nicola Dell, Damon McCoy, and Thomas Ristenpart. 2018. The Spyware Used in Intimate Partner Violence. In *IEEE Symposium on Security and Privacy*. IEEE, 441–458. https://doi.org/10.1109/SP.2018.00061

[13] Gang Chen, Yangqiu Song, Fei Wang, and Changshui Zhang. 2008. Semi-supervised multi-label learning by solving a sylvester equation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, 410–419.

[14] Hongge Chen, Huan Zhang, Duane Boning, and Cho-Jui Hsieh. 2019. Robust decision trees against adversarial examples. In *International Conference on Machine Learning*. PMLR, 1122–1131.

[15] Danielle Keats Citron. 2015. Spying Inc. *Washington and Lee Law Review* 72, 3 (2015), 1242–1282.

[16] Coalition Against Stalkerware. 2020. What is Stalkerware. https://stopstalkerware.org/#whatis.

[17] Shubhomoy Das, Weng-Keen Wong, Thomas Dietterich, Alan Fern, and Andrew Emmott. 2016. Incorporating Expert Feedback into Active Anomaly Discovery. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 853–858. https://doi.org/10.1109/ICDM.2016.0102

[18] Anthony Desnos and Patrik Lantz. 2011. Droidbox: An android application sandbox for dynamic analysis. *Lund Univ., Lund, Sweden, Tech. Rep* (2011).

[19] Karen M Devries, Joelle Y Mak, Loraine J Bacchus, Jennifer C Child, Gail Falder, Max Petzold, Jill Astbury, and Charlotte H Watts. 2013. Intimate partner violence and incident depressive symptoms and suicide attempts: a systematic review of longitudinal studies. *PLoS Med* 10, 5 (2013), e1001439.

[20] Jill P. Dimond, Casey Fiesler, and Amy S. Bruckman. 2011. Domestic violence and information communication technologies. *Interacting with Computers* 23, 5 (2011), 413–421.

[21] Manuel Egele, Chris Krügel, Engin Kirda, Heng Yin, and Dawn Song. 2007. Dynamic Spyware Analysis. In *USENIX Annual Technical Conference*.

[22] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. 2014. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)* 32, 2 (2014), 1–29.

[23] Jerry Finn. 2004. A survey of online harassment at a university campus. *Journal of Interpersonal violence* 19, 4 (2004), 468–483.

[24] Diana Freed, Sam Havron, Emily Tseng, Andrea Gallardo, Rahul Chatterjee, Thomas Ristenpart, and Nicola Dell. 2019. "Is My Phone Hacked?" Analyzing Clinical Computer Security Interventions with Survivors of Intimate Partner Violence. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 202 (Nov. 2019), 24 pages. https://doi.org/10.1145/3359304

[25] D. Freed, Jackeline Palmer, Diana Elizabeth Minchala, K. Levy, T. Ristenpart, and Nicola Dell. 2018. "A Stalker's Paradise": How Intimate Partner Abusers Exploit Technology. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (2018).

[26] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely Randomized Trees. *Mach. Learn.* 63, 1 (April 2006), 3–42. https://doi.org/10.1007/s10994-006-6226-1

[27] Jacqueline M. Golding. 1999. Intimate Partner Violence as a Risk Factor for Mental Disorders: A Meta-Analysis. *Journal of Family Violence* 14 (1999), 99–132.

[28] Google. 2020. Developer Program Policy: September 16, 2020 announcement. https://support.google.com/googleplay/android-developer/answer/10065487#stalkerware.

[29] D. Harkin and Adám Molnár. 2021. Operating-System Design and Its Implications for Victims of Family Violence: The Comparative Threat of Smart Phone Spyware for Android Versus iPhone Users. *Violence Against Women* 27 (2021), 851 – 875.

[30] Laura Hautala. 2020. Stalkerware sees all, and US laws haven't stopped its spread. Installing hidden spy software is illegal, so why is it so easy? https://www.cnet.com/news/stalkerware-sees-all-and-us-laws-havent-stopped-its-spread/.

[31] Engin Kirda, Christopher Kruegel, Greg Banks, Giovanni Vigna, and Richard Kemmerer. 2006. Behavior-based Spyware Detection.. In *Usenix Security Symposium*. 694.

[32] Klaus Krippendorff. 2018. *Content analysis: An introduction to its methodology*. Sage publications.

[33] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. 2017. *Research methods in human-computer interaction*. Morgan Kaufmann.

[34] Karen Levy and Bruce Schneier. 2020. Privacy threats in intimate relationships. *Journal of Cybersecurity* 6, 1 (05 2020). https://doi.org/10.1093/cybsec/tyaa006 arXiv:https://academic.oup.com/cybersecurity/article-pdf/6/1/tyaa006/33328242/tyaa006.pdf tyaa006.

[35] Peter Mertin and Philip Mohr. 2002. A follow-up study of posttraumatic stress disorder, anxiety, and depression in Australian victims of domestic violence. *Violence and Victims* 16, 6 (January 2002), 645–654.

[36] National Network to End Domestic Violence (NNEDV). 2019. Spyware and Stalkerware: Phone Surveillance & Safety for Survivors. Tech Safety https://www.techsafety.org/s/NNEDV_Spyware_Phones_2019-ywrd.pdf.

[37] Organization for Security and Cooperation (OSCE). 2019. *OSCE-led Survey on Violence Against Women*. Technical Report.

[38] Christopher Parsons, Adam Molnar, Jakub Dalek, Jeffrey Knockel, Miles Kenyon, Bennett Haselton, Cynthia Khoo, and Ronald Deibert. 2019. The Predator in Your Pocket. A Multidisciplinary Assessment of the Stalkerware Application Industry. Research Report 119 https://citizenlab.ca/docs/stalkerware-holistic.pdf.

[39] Jessica A Pater, Moon K Kim, Elizabeth D Mynatt, and Casey Fiesler. 2016. Characterizations of Online Harassment: Comparing Policies Across Social Media Platforms. In *Proceedings of the 19th International Conference on Supporting Group Work*. 369–374. https://doi.org/10.1145/2957276.2957297

[40] Kevin Alejandro Roundy, Paula Barmaimon Mendelberg, Nicola Dell, Damon McCoy, Daniel Nissani, Thomas Ristenpart, and Acar Tamersoy. 2020. The Many Kinds of Creepware Used for Interpersonal Attacks. In *IEEE Symposium on Security and Privacy*. IEEE. https://doi.org/10.1109/sp40000.2020.00069

[41] Neil Rubens, Mehdi Elahi, Masashi Sugiyama, and Dain Kaplan. 2015. *Active Learning in Recommender Systems*. Springer US, Boston, MA, 809–846. https://doi.org/10.1007/978-1-4899-7637-6_24

[42] Leslie A Sackett and Daniel G Saunders. 1999. The impact of different forms of psychological abuse on battered women. *Violence and victims* 14, 1 (1999), 105–117.

[43] Nithya Sambasivan, Amna Batool, Nova Ahmed, Tara Matthews, Kurt Thomas, Laura Sanely Gaytán-Lugo, David Nemer, Elie Bursztein, Elizabeth Churchill, and Sunny Consolvo. 2019. " They Don't Leave Us Alone Anywhere We Go" Gender and Digital Abuse in South Asia. In *proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–14.

[44] Stefan Saroiu, Steven D. Gribble, and Henry M. Levy. 2004. Measurement and Analysis of Spyware in a University Environment. In *First Symposium on Networked Systems Design and Implementation (NSDI 04)*. USENIX Association, San Francisco, CA. https://www.usenix.org/conference/nsdi-04/measurement-and-analysis-spyware-university-environment

[45] Burr Settles. 2010. Active Learning Literature Survey. *Computer Sciences Technical Report 1648* (2010).
[46] Y. Shen, Pierre-Antoine Vervier, and G. Stringhini. 2021. Understanding Worldwide Private Information Collection on Android. *ArXiv* abs/2102.12869 (2021).
[47] Wissam Siblini, Pascale Kuntz, and Frank Meyer. 2018. CRAFTML, an Efficient Clustering-based Random Forest for Extreme Multi-label Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 4664–4673. http://proceedings.mlr.press/v80/siblini18a.html
[48] Heidi Stöckl, Karen Devries, Alexandra Rotstein, Naeemah Abrahams, Jacquelyn Campbell, Charlotte Watts, and Claudia Garcia Moreno. 2013. The global prevalence of intimate partner homicide: a systematic review. *The Lancet* 382, 9895 (2013), 859–865. https://doi.org/10.1016/S0140-6736(13)61030-2
[49] Kurt Thomas, Devdatta Akhawe, Michael Bailey, Dan Boneh, Elie Bursztein, Sunny Consolvo, Nicola Dell, Zakir Durumeric, Patrick Gage Kelley, Deepak Kumar, Damon McCoy, Sarah Meiklejohn, Thomas Ristenpart, and Gianluca Stringhini. 2021. SoK: Hate, Harassment, and the Changing Landscape of Online Abuse. In *2021 2021 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 473–493. https://doi.org/10.1109/SP40001.2021.00028
[50] Jenna Torluemke and Christine Kim. 2020. Nearly Half of Americans Admit to 'Stalking' an Ex or Current Partner Online. NortonLifeLock Press Release https://www.businesswire.com/news/home/20200212005192/en/.
[51] Emily Tseng, Rosanna Bellini, Nora McDonald, Matan Danos, R. Greenstadt, Damon McCoy, Nicola Dell, and T. Ristenpart. 2020. The Tools and Tactics Used in Intimate Partner Surveillance: An Analysis of Online Infidelity Forums. In *USENIX Security Symposium*.
[52] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2010. Mining Multi-label Data. In *Data Mining and Knowledge Discovery Handbook*, Oded Maimon and Lior Rokach (Eds.). Springer US, Boston, MA, 667–685. https://doi.org/10.1007/978-0-387-09823-4_34
[53] Changhu Wang, Shuicheng Yan, Lei Zhang, and Hongjiang Zhang. 2009. Multi-label sparse coding for automatic image annotation. In *CVPR*. 1643–1650.
[54] Hao Wang, Somesh Jha, and Vinod Ganapathy. 2006. NetSpy: Automatic Generation of Spyware Signatures for NIDS. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*. 99–108. https://doi.org/10.1109/ACSAC.2006.34
[55] Liantao Wang, Xuelei Hu, Bo Yuan, and Jianfeng Lu. 2015. Active learning via query synthesis and nearest neighbour search. *Neurocomputing* 147 (2015), 426–434. https://doi.org/10.1016/j.neucom.2014.06.042 Advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012).
[56] Christopher K. I. Williams. 1998. Computation with Infinite Neural Networks. *Neural Computation* 10, 5 (1998), 1203–1216. https://doi.org/10.1162/089976698300017412
[57] Lok Kwong Yan and Heng Yin. 2012. Droidscope: Seamlessly reconstructing the {OS} and dalvik semantic views for dynamic android malware analysis. In *21st {USENIX} Security Symposium ({USENIX} Security 12)*. 569–584.
[58] Bishan Yang, Jian-Tao Sun, Tengjiao Wang, and Zheng Chen. 2009. Effective Multi-Label Active Learning for Text Classification. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Paris, France) *(KDD '09)*. Association for Computing Machinery, New York, NY, USA, 917–926. https://doi.org/10.1145/1557019.1557119
[59] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit S. Dhillon. 2014. Large-scale Multi-label Learning with Missing Labels. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32* (Beijing, China) *(ICML'14)*. JMLR.org, I–593–I–601. http://dl.acm.org/citation.cfm?id=3044805.3044873
[60] Yin Zhang Yu-Yin Sun and Zhi-Hua Zhou. 2010. Multi-Label Learning with Weak Label. In *AAAI*. 593–598.
[61] Min-Ling Zhang and Zhi-Hua Zhou. 2014. A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (2014), 1819–1837. https://doi.org/10.1109/TKDE.2013.39
[62] Dengyong Zhou, Bernhard Schölkopf, and Thomas Hofmann. 2004. Semi-Supervised Learning on Directed Graphs. In *Proceedings of the 17th International Conference on Neural Information Processing Systems* (Vancouver, British Columbia, Canada) *(NIPS'04)*. MIT Press, Cambridge, MA, USA, 1633–1640.
[63] Shenghuo Zhu, Xiang Ji, Wei Xu, and Yihong Gong. 2005. Multi-Labelled Classification Using Maximum Entropy Method. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Salvador, Brazil) *(SIGIR '05)*. Association for Computing Machinery, New York, NY, USA, 274–281. https://doi.org/10.1145/1076034.1076082
[64] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning* (Washington, DC, USA) *(ICML'03)*. AAAI Press, 912–919.

A APPENDIX: CONCRETE EXAMPLE OF DOSMELT'S OUTPUT

As discussed in Section 6.4, we illustrate 19 stalkerware apps detected by Dosmelt in Figure.5, when we deploy Dosmelt in practices. We also show the identified surveillance capabilities of these apps produced from Dosmelt. These surveillance capabilities are then verified and confirmed by the human analysts in the team. The results of the nuanced surveillance type attribution provide a concrete example of the precise warning information that can be extracted from Dosmelt's output. As observed, the first two columns in Figure.5 are the ids and the titles of the detected stalkerware apps. Each column from the 3rd column to the 16th column corresponds to each one of the 14 surveillance types listed in our taxonomy (seen in Table.1 of Section.3). With this setting, each row in Figure.5 shows all the surveillance capabilities that one detected stalkerware app conducts, as identified by Dosmelt and then verified by the human experts. The 1 and 0 given in each row of Figure.5 denote whether a stalkerware app is equipped with a specific surveillance capability or not. For example, the app with the id *com.celphtr.rodes* and titled as *Cell Phone Tracker* has 5 different surveillance functions at the same time: *SMS*, *GPS-Tracking*, *Social-Media*, *Call-Logs* and *Contacts*. From the figure, we can have an intuitive understanding about the multi-label nature of the nuanced surveillance type attribution problem tackled in our study. A stalkerware app can spy on the targeted device/victim using a variety of surveillance functionalities. We use the same column setting in the our stalkerware data set uploaded to the Coalition Against Stalkerware's *Stalkerware Threat List*.

App ID	App Title	Browsing-History	Call-Logs	Call-Recordings	Camera	Contacts	Email	GPS-Tracking	Installed-Apps	Keylogging	Microphone	Photo Extraction	Screen	SMS	Social-Media
com.celphtr.rodes	Cell Phone Tracker	0	1	0	0	1	0	1	0	0	0	0	0	1	1
child.monitor.app	Couple Monitor -Mobile Tracker	1	1	0	0	1	0	1	0	0	0	0	0	1	0
project.antitheft	Telesom Antitheft	0	1	0	1	0	0	1	0	0	1	0	0	0	0
com.trphwhat.prob	Call Tracker	0	1	0	0	1	0	1	0	0	0	0	0	1	0
com.viva.recoverymyfileprank	Recover My Files PRO	0	0	0	0	1	0	0	0	0	0	1	0	1	1
com.calltracker.calltracker	Cell Tracker	1	0	0	0	1	0	1	0	0	0	0	0	1	0
find.my.device.tracking	find my phone tracking pro	0	1	0	0	1	0	1	0	0	0	0	0	0	0
com.appmartspace.eazytracker	Easy Tracker	0	1	0	0	0	0	1	0	0	0	0	0	1	0
com.restore.backup.free.prov	Watsup Recova	1	0	0	0	0	0	0	0	0	0	1	0	0	1
com.track.lost.cell.phone.lite.lost.device.tracker.lite	Track Lost Cell Phone: Lost Device Tracker Lite	0	0	0	1	0	0	1	0	0	0	0	1	0	0
khabarizone.mobilenumberlocationtracker	Mobile Number Location Tracker	0	1	0	0	0	0	1	0	0	0	0	0	0	0
com.dubaigamesstudio.voicecallrecorderfree	Voice Call Recorder - Free	0	1	1	0	0	0	0	0	0	0	0	0	0	0
com.internaliagroup.seguridad360	Mobile Security 360	0	0	0	0	0	0	1	1	0	0	0	0	0	0
com.picturesrecovery.restorefilesfree	Recover All Deleted Photos:Files,Images	0	0	0	0	1	0	0	0	0	0	1	0	0	0
gbwhatsaap.aplijmz	GBwhatsaap	0	0	0	0	0	0	0	0	0	0	0	0	0	1
com.geotou.findmyfamily	Find My Friends	0	0	0	0	0	0	1	0	0	0	0	0	0	0
com.automaticcallrecorder2016free.callrecorderpro	Automatic Call Recorder - Free	0	0	1	0	0	0	0	0	0	0	0	0	0	0
com.octadata.videorecover	Recover Video	0	0	0	0	0	0	0	0	0	0	1	0	0	0
com.Mob123.lzen456	Recorder tips Screen Record Capture	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 5: The examples of stalkerware detection and nuanced surveillance attribution results of DOSMELT