The Many Kinds of Creepware Used for Interpersonal Attacks

Kevin A. Roundy*, Paula Bermaimon Mendelberg†, Nicola Dell†, Damon McCoy‡, Daniel Nissani†, Thomas Ristenpart†, Acar Tamersoy*

*NortonLifeLock Research Group †Cornell Tech ‡New York University

***Abstract*—Technology increasingly facilitates interpersonal attacks such as stalking, abuse, and other forms of harassment. While prior studies have examined the ecosystem of software designed for stalking, there exists an unstudied, larger landscape of apps—what we call creepware—used for interpersonal attacks. In this paper, we initiate a study of creepware using access to a dataset detailing the mobile apps installed on over 50 million Android devices. We develop a new algorithm, CreepRank, that uses the principle of guilt by association to help surface previously unknown examples of creepware, which we then characterize through a combination of quantitative and qualitative methods. We discovered apps used for harassment, impersonation, fraud, information theft, concealment, and even apps that purport to defend victims against such threats. As a result of our work, the Google Play Store has already removed hundreds of apps for policy violations. More broadly, our findings and techniques improve understanding of the creepware ecosystem, and will inform future efforts that aim to mitigate interpersonal attacks.**

I. Introduction

Technology is increasingly used as a vector for interpersonal attacks. One prominent example is in intimate partner violence (IPV), where victims report abusers utilizing apps for a range of harms, including text message "bombing" (sending hundreds or thousands of messages), spoofing phone numbers to hide the source of harassment, creating fake suggestive images to hurt a victim's reputation, and installing spyware apps on victim devices [1]–[4]. Only the last category has been studied: Chatterjee et al. [5] performed measurements on official app stores and the web more broadly to discover a large number of surveillance apps advertised to, and easily used by, abusers. However, there has been no exploration of the broader landscape of software enabling the many other forms of harassment reported by victims.

This paper describes the first measurement study aimed at illuminating the broader ecosystem of what we call ***creepware***: apps whose primary use case is enabling non-expert users to mount interpersonal attacks. Apps only sometimes used for harassment (e.g., email or messaging apps) fall outside our purview. We find that the ecosystem surrounding creepware also includes apps advertising the ability to defend against interpersonal attacks, which we study in order to provide a more holistic understanding of this problem space.

Unfortunately, the prior techniques [5] used to study spyware are not helpful here. They rely on knowledge of spyware-specific search terms, whereas a priori we do not know what types of creepware apps people seek out. Instead, we turn to the principle of guilt by association, which has previously been used to discover new strains of conventional malware [6]–[8]. The key idea is that software that disproportionately appears on the same device as known malware is, itself, likely to be malicious. However, adapting such an approach to the creepware context requires large amounts of data about app installations and new algorithms.

We partnered with Norton, a major computer security firm, to obtain anonymized data about billions of app installations on 50 million Android devices over several years protected by Norton Mobile Security.[1] We couple this data with a new algorithm, CreepRank, that, given a set of seed apps known to be creepware, assigns scores to other apps. At its core, CreepRank is a graph mining algorithm that computes scores using maximum a posteriori estimation, which helps suppress false positives among rare apps (a problem that similar algorithms face in this context when not using a skeptical prior, as we shall see). Intuitively, the higher the CreepRank, the more the app is associated, via co-installation data, with known creepware.

We applied CreepRank to the Norton dataset using as seed set the overt spyware surveillance apps identified by Chatterjee et al. [5]. The resulting ranking helped us discover a wide variety of potential creepware apps. To make sense of these results, we manually coded the 1,000 apps with highest CreepRank. This involved iteratively developing a new taxonomy of interpersonal attack and defense apps. Coders used the app title, package name (app ID), description (when available), and additional metadata, such as installation counts, to label each app with a code from our taxonomy.

The findings from our manual coding analysis showed that 857 of CreepRank's top 1,000 apps qualify as creepware, fulfilling a clear purpose pertaining to interpersonal attack or defense. Unsurprisingly, given the seed set, surveillance apps were best represented in the rankings—372 of the top 1,000 apps—many of which were not identified by prior work. Among these are 107 multifaceted surveillance apps that affected 172 K Norton customers in 2017 alone. Overall, CreepRank identified more than a million installs of diverse creepware apps, including apps that enable spoofing (114 apps, see an example in Figure 1), harassment (80, including SMS bombers), hacking tutorials (63), and many more. We also

[1]We received IRB approval for our study.

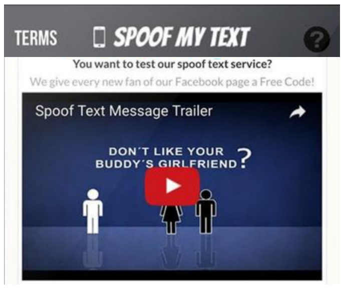

TERMS
SPOOF MY TEXT
You want to test our spoof text service?
We give every new fan of our Facebook page a Free Code!
Spoof Text Message Trailer
DON'T LIKE YOUR BUDDY'S GIRLFRIEND?

Fig. 1: The *Spoof Text Message* app advertises with a video whose opening lines are: “Don’t like your buddy’s girlfriend? Well, break them up!” [13].

found apps that aim to defend against interpersonal attacks, such as anti-surveillance apps and apps that deal with SMS bombing. Moreover, even the 143 apps surfaced by CreepRank that did not have a clear attack or defense purpose fit into interesting trends that yield insights into the inclinations of creepware users. We further explore CreepRank’s utility by seeding it with different sets of attack apps, investigating creepware trends over time, and more.

Our findings suggest that CreepRank is a valuable tool for understanding the ecosystem of apps used for interpersonal attack and defense. It is also practically useful. CreepRank-identified apps now trigger warnings in Norton’s products and are flagged as potentially dangerous apps when scanning phones of IPV survivors in the context of Cornell Tech’s computer security clinic [9], [10]. We also reported 1,095 apps to Google via a responsible disclosure process, and they removed 813 apps for violating the Google Play store’s terms and conditions.

In summary, our contributions include the following:

- We introduce CreepRank, an algorithm that leverages the principle of guilt by association to discover creepware. We show that it measures creepware effectively: it identifies 2.8x more creepware among its top 1,000 apps than random walk with restart [11], [12], another well-known graph-based algorithm. We use CreepRank to drive the first measurement study of the creepware ecosystem.
- Using manual coding of 1,000 CreepRank-identified apps, we discover new classes of creepware and develop a creepware taxonomy that should be broadly useful.
- We explore use of CreepRank with distinct seed sets, study trends in creepware over time, and analyze types of creepware commonly found together on devices.

II. Background and Related Work

Apps used for interpersonal attacks. Our paper contributes to a small but growing body of work that seeks to understand the role of technology in interpersonal attacks, such as those that arise in IPV [1]–[3], technology-facilitated bullying [14], [15], and other forms of targeted harassment. In particular, our work builds on research by Chatterjee et al. [5] that first highlighted the types of apps being used as spyware in IPV. Their research discovered spyware apps by crawling forums and app marketplaces for candidate apps, via search terms typically used to find spyware. They highlight the tricky issue of dual-use apps, or apps that may have a legitimate purpose but are often easily re-purposed for abuse.

While we also investigate (previously unknown) spyware apps used in IPV, our focus is broader. We do not restrict attention to IPV use cases, but instead consider interpersonal attacks in general. We also want to understand apps beyond spyware that enable harassment, SMS-bombing, spoofing (e.g., Figure 1), and more. Although prior qualitative work interviewing victims of technology-enabled abuse [1]–[3] indicated that abusers use some of these categories of apps, no research has been done to measure or characterize them. We refer to this broader class of apps as ***creepware***, defined for our purposes as apps for which a predominant use case is enabling non-technology-expert users to mount interpersonal attacks.

This explicitly leaves out of scope some classes of apps. We do not consider apps which are *not* intended or predominantly used for interpersonal attacks, such as popular email, text messaging, and social media apps. While these are often vectors for abuse, the vast majority of use is benign and mitigation requires different approaches than for creepware. We also do not consider some malicious apps that require more expertise to obtain and use, such as remote access trojans (RATs) and other malware used by governments or voyeurs, which have been investigated in prior work [16]–[18].

Potentially unwanted programs (PUPs) [19], [20] are usually commercially-motivated malware that exploit pay-per-install services to add “bloatware” to a device for purposes of financial profit. We did encounter examples of pay-per-install apps [21] in our study, as discussed in Section VI. Prior studies have also investigated more malicious types of malware that directly steal user secrets (e.g., bank details) [22]–[27]. However, PUPs and such malware differ from creepware-type interpersonal attack apps in that the formers’ authors seek to have them distributed broadly and indiscriminately, rather than being deliberately installed by one person to attack another.

Using app installation information. Our approach for discovering creepware apps is based on the principle of ***guilt by association (GBA)****: interpersonal attack and defense apps are disproportionately installed on the same devices.* By disproportionately, we mean in relation to the likelihood that these apps are co-installed with other kinds of apps. We were inspired by Polonium [28] and other systems [6], [7] that use the GBA principle to identify PC malware. By representing software installation data as a graph of software files and computers, Polonium initializes node weights, using domain knowledge and an extensive set of ground-truth benign and malicious software, and then applies the belief propagation (BP) [29] algorithm. BP treats nodes as random variables with at least two states (e.g., good and bad) and produces the marginal probability distribution for each node in the graph over these states. Unfortunately, BP requires labeled data, making it ill-suited to our single state/class setting in which only a few apps are known to be used for inter-personal attacks (e.g., because they advertise as such), and for which there is no obvious way to construct a representative set of benign apps known to be unusable for such attacks.

Other algorithms may be more suitable for our task, such as random walk with restart (RWR) [11], [12], which only requires a small set of ground-truth labels for one class. Although RWR is an exploratory method, we found it ill-suited to our task because it assigns high scores to rare apps that are installed alongside interpersonal attack apps due to random chance. RWR lacks a way to add a skeptical prior belief, i.e., to assume that apps are likely innocent until proven guilty by numerous associations with "guilty" apps. Our need to incorporate prior beliefs into an exploratory GBA algorithm led us to design CreepRank, and proves to be its most important characteristic, as discussed in Section VI. While the techniques we develop for CreepRank may prove useful for exploring other classes of apps, we have only investigated its utility for discovering interpersonal attack and defense apps.

Finally, prior work has also explored what can be learned from the apps installed on a device [30] or the set of apps used at least once a month [31], including predicting a device owner's demographic information (e.g., gender). We explore how the combination of creepware (and/or creepware defense) apps installed on a device might point to user behaviors, such as credit card fraud or interpersonal abuse (see Section VII-C).

III. Dataset Description and Properties

We develop new data-driven approaches for discovering apps used in interpersonal attack and defense that leverage datasets consisting of anonymized Android app installations recorded by NortonLifeLock's Norton Mobile Security app. For each device in our datasets, we have a list of <*package name, relative time*> tuples reflecting the apps that were installed on the device. The package name (Android app ID) used to register apps in the Google Play store (if it has ever been distributed there) is extracted from each APK file. For apps exclusively distributed off-store, the app ID need not be registered, and is therefore not necessarily a unique identifier for an app. As a result, it is likely that our methods are most effective for discovering abusive on-store apps, though we have found that fixed package names are common for off-store apps and that polymorphic package names are rare in practice.

The relative time of each app installation is derived from the time at which it was first scanned by the Norton app. Thus, the relative time generally has a value close to 0 for all apps installed prior to Norton's app and, for subsequently installed apps, indicates time relative to the installation of the Norton app. The dataset does not include information on if and when apps were removed from a device.

We use two different datasets: (1) data gathered from devices active in calendar year 2017 and (2) data gathered over a year-long period from May 1, 2018 to May 1, 2019. We refer to these as the 2017 and 2018/2019 datasets, respectively. The 2017 dataset includes 27.7 million devices with 10.9 million unique package names that were installed around 4 billion times across the devices (not counting duplicate app installations and app updates). The 2018/2019 dataset has 22.6 million devices with 7.5 million unique package names that were installed 1.9 billion times. The datasets are not disjoint, 4.5 million devices appear in both datasets.

We also use a dataset of marketplace data provided by Norton that was periodically scraped from the Google Play store over a period of several years. While this dataset is missing data from some apps, it provides good coverage of apps that have been retired or forcibly removed from the app store and its website. For each app, it includes its genre, title, description, and permissions.

Our data also has limitations. The devices in the dataset are not necessarily representative of typical users, as by definition they have Norton's security app installed and are therefore security conscious. For example, many IPV victims face financial challenges [32] (Norton's security app is not free) and have limited awareness of digital security [1]. We do investigate, within the limits of the data, when and (seemingly) why the Norton app is used in relation to the types of interpersonal attack apps found on a device (Section VII-D).

In addition, any dataset of this nature includes devices that do not represent normal use. An example is devices used by anti-virus (AV) testers and researchers, on which many malicious apps will appear. To limit the impact of such abnormalities, we removed from the dataset all devices on which more than 1,000 apps were installed during the course of one year, as these seem unlikely to represent real users' devices. This removed about 18 thousand devices from the 2017 dataset and 9 thousand from the 2018/2019 dataset.

Finally, to make the dataset more manageable, we excluded the top 1.1% most prevalent apps and then dropped devices with only one app installation. These apps are likely benign and are not interesting for our purposes, and would interfere with the efficacy of our algorithms. This reduced the number of app installations in the 2017 data from 4 billion to 546 million, for 10.8 million apps on 25 million devices. App installations in the 2018/2019 dataset dropped from 1.9 billion to 205 million, for 7.4 million apps on 17 million devices.

IV. Using Guilt by Association for App Discovery

We are interested in apps useful for interpersonal attack and defense. These include apps that are used by one person to monitor, harass, or otherwise harm another person (attack), apps used to prevent such attacks (defense), and apps that are useful for both attack and defense. The first category is what we refer to as creepware. This paper will surface classes of interpersonal attack and defense apps that were completely new to the authors (and, we suspect, many others). We will see many examples in subsequent sections.

Our hypothesis for discovering such apps is based on the principle of guilt by association (GBA), which infers that apps that tend to be installed on devices infected by malicious apps tend to be malicious themselves. For an initial assessment of

Device 1	Device 2
Track a Phone by Number	GirlFriend Cell Tracker
Find My Friends	*System Services (aka mSpy)*
Live Mobile Location Tracker	Hidden Auto Call Recorder
SMS from PC / Tablet... Sync	Family Locator - GPS Tracking
HelloSpy	SMS Forwarder

Table I: Surveillance apps co-installed with a known covert surveillance app (shown in italics), ordered by install time.

the viability of the GBA hypothesis, we searched for apps that use the keywords "spy" or "track" in their title or package name on the 35,811 devices infected by one or more of 18 off-store intimate partner surveillance apps identified by Chatterjee et al. [5]. We found many devices with multiple surveillance-style apps installed. The titles of surveillance apps installed on two representative devices are shown in Table I. This gave us hope that GBA would be useful more broadly.

The GBA hypothesis has two main benefits for identifying and measuring creepware. First, guilt spreads naturally from one class of abusive app to another even when they are very different, spreading even to defensive apps that counter abuse. Second, GBA necessarily identifies apps that are actually installed by abusers and/or victims in practice, as it is based entirely on co-installation data and does not use app properties or metadata (in contrast to [5]).

We now turn to developing an exploratory graph mining algorithm that exploits the GBA hypothesis. The algorithm, CreepRank, takes as input a set of seed apps, an installation dataset, and outputs a ranking for each app in the dataset. When seeded with known creepware apps, high-ranking apps are likely to be associated with interpersonal attack or defense. This section proceeds by describing seed set selection for CreepRank, its use of first-order correlations among apps, its false-positive mitigation scheme, and finally, the method whereby it captures high-order correlation among apps.

A. Seed Set Selection

Our method is fundamentally a one-class algorithm in that it measures the relevance between a focused set of seed apps and all other apps. No other labeled data is required. Our examination and coding of creepware apps is based on a seed set of 18 overt surveillance apps identified by Chatterjee et al. [5] that openly market themselves as usable for intimate partner surveillance purposes, and which are sold outside of the Google Play app marketplace because they do not conform to marketplace rules. In Section VII-B we experiment with CreepRank's ability to explore narrower ecosystems by seeding it with a variety of different seed sets.

B. First-Order Graph Algorithm

We determined that the most direct way to leverage the GBA principle for mobile apps was to estimate the frequency with which each app appears on a device that has been infected with a seed set app. We start by representing installation data as a graph, in which we represent apps and devices as nodes, and add edges to represent the installation of apps on devices.

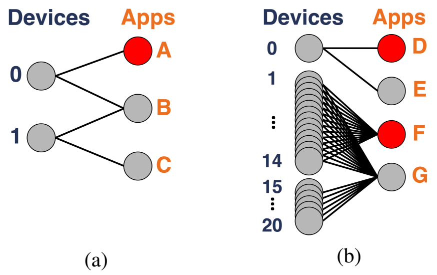


Fig. 2: Example bipartite graph representations of app installation data. An edge represents installation of the app on the device. Seed set nodes are shown in red.

In Figure 2a, for example, app A is a seed set app installed on Device 0, while app B is installed on Devices 0 and 1, one of which is infected by a seed set app, while the other is not.

Formally, we model k, the number of infected devices on which an app appears out of n total devices as a random variable X drawn from a binomial distribution $B(n, p)$, such that $P(X = k|p) = \binom{n}{k} p^k (1-p)^{n-k}$, where p denotes the probability that the app appears on an infected device. Then, probability p can be readily estimated from the installation data using maximum likelihood estimation (MLE), which yields $p = k/n$. Thus, the MLE method would estimate the probability with which an app appears on an infected device by dividing its observed installations on infected devices by its total observed installations, which can then be used as a risk score for the app and to rank all unknown apps.

C. Adding False Positive Suppression

While the MLE method of probability estimation is appealingly simple, when applied to our data to rank creepware, it suffers from high false positive (FP) rates. To understand why, consider apps E and G in Figure 2b. App E appears on only one device, which is infected by app D, so MLE outputs $p_E = 1/1$, whereas for app G, which is on 14 infected devices, it returns $p_G = 14/20$. When we consider that the dataset contains observations about more than 10 million apps, nearly all of which are benign, app G is intuitively more suspicious than the millions of rare apps like app E, whose sole instance could have appeared on an infected device by random chance. This intuition was born out in practice; our attempts to apply the MLE method as a ranking tool yielded low quality rankings with many irrelevant apps, as described in Section VI-A.

CreepRank therefore uses maximum a posteriori (MAP) probability estimates, which are similar to MLE's optimization method, except the a posteriori estimates incorporate the random variable's prior probability into the maximization objective. That is, the MAP method incorporates an estimate of the prior probability that apps appear on infected devices and applies Bayes' rule to choose the parameters of the posterior probability distribution that maximize the probability of the observed data given knowledge of the prior.

We must therefore estimate the prior probability distribution with which apps appear on infected devices, which we do by

CreepRank

```
Input: Per-device edge lists of installed apps and list of seed set apps
1: Initialize seed-set apps with app_score=1, otherwise app_score=0
2: Set each dev_score = max(app_scores) across connected apps
3: Set each app_score = avg(dev_score)
4: Normalize app_scores
5: If not converged then goto step 2
6: Apply MAP formula to obtain final app_scores
Output: Apps ranked in decreasing order of app_scores
```

Fig. 3: The CreepRank algorithm to capture high-order correlations between apps and devices they are co-installed on.

	Device Scores		Normalized App Scores		
Round	0	1	A	B	C
0	–	–	1	0	0
1	1	0	1	0.5	0
2	1	0.5	1	0.75	0.5
3	1	0.3	1	0.65	0.3
4	1	0.342	1	0.671	0.342
5	1	0.331	1	0.666	0.331
6	1	0.334	1	0.667	0.333
7	1	0.333	1	0.667	0.333

Table II: CreepRank applied to the graph of Figure 2a, showing convergence to 3 significant digits by the 7th iteration.

applying the MLE method from Section IV-B to all apps that appear on at least 100 devices (we do not include rare apps as these may produce unreliable MLE values). We then model the prior probability distribution as a beta distribution Beta(α, β) that we fit to our MLE values, obtaining $\alpha = 1.09$ and $\beta = 186$. Our use of a beta distribution to model the prior is convenient, as the beta distribution is a conjugate prior for the binomial distribution with which we model our observations, meaning that the posterior probability distribution is also a beta distribution, with parameters Beta$(\alpha + k, \beta + n - k)$, for which the MAP estimate of the mode of this distribution is readily derived as $(k + \alpha - 1)/(n + \alpha + \beta - 2)$.

Note that for our prior of Beta$(1.09, 186)$, the MAP estimates contrast with MLE primarily by adding a large constant to the denominator of the estimate. Practically, this means that the MAP estimate assigns small CreepRank values to apps that are not observed on infected devices in large numbers, but the effect of the prior diminishes as k and n increase.

D. Capturing High-Order Correlations Among Apps

The final component of CreepRank reduces the algorithm's sensitivity to the small seed sets for which it is designed by enabling it to capture high-order correlations between the seed set and the broader ecosystem of creepware apps. We considered alternative high-order graph-based methods, such as random walk with restart (RWR) [11], [12], which provides no mechanism to suppress false positives among rare apps, causing it to include many irrelevant apps.

The steps of our algorithm, CreepRank, are shown in Figure 3. Iterative application of these steps to the graph shown in Figure 2a results in the values shown in Table II. The input to CreepRank is a list of the apps installed on each device, and a list of seed set apps. From this input, a bipartite graph between device and app nodes is constructed, with edges indicating an app's presence on a particular device. CreepRank's first step initializes the seed set apps with score 1 and all other apps with score 0. In Step 2, each device receives an infection score that is the maximum value of all apps installed on the device (these scores are binary in the first iteration). In Step 3, apps are assigned a score based on the average score of the devices on which the app appears, and the scores are normalized in Step 4 to ensure that the sum of the app scores is equal to the sum of all MLE values obtained by the first-order method described in Section IV-B. In the absence of normalization, the max function applied in Step 2 would cause app scores to increase with each iteration of the algorithm. Any desired convergence criteria can be set for Step 5. We ran our algorithm for 10 iterations, since by then the rankings became stable even for graphs of 500 million edges. For instance, the maximum score delta was .00085 in the algorithm's 10th iteration for 2017 data, for a seed set of 18 apps (scores range from 0 to 1).

E. Implementation

Our datasets are quite large. For example, the graph corresponding to the 2017 installation dataset consists of 546 million edges, 25 million device nodes, and 10.6 million app nodes. We therefore implemented CreepRank for use in a distributed setting. The algorithm required only 77 lines of code, which consist of 29 Scala commands that make ample use of Spark. The algorithm ran on 100 Spark worker nodes on an AWS cluster, each node consisting of a single CPU core and 10GB of RAM, plus a driver node with 15GB of RAM. These workers ran on a mix of AWS instances of type m5.12xlarge (48 cores and 192GB RAM) and r5.4xlarge (16 cores and 128GB RAM). The average execution time, taken over 10 executions of CreepRank on the 2017 dataset, was 24 minutes, 21 seconds with a standard deviation of 115 seconds. Writing out the ranking scores of all 10 million apps to a Hadoop File System takes an additional 90 seconds.

V. Categorizing Creepware

After running CreepRank on 2017 data with a seed set of 18 covert surveillance apps, we wanted to characterize the categories of apps discovered. To achieve this, we manually coded 1,000 apps that (1) were highest ranked by the algorithm and therefore most risky, and (2) had at some point been available on the Google Play store and for which we could therefore obtain sufficiently detailed data via Internet searches. The overarching question we sought to answer was: *What categories of creepware exist beyond interpersonal surveillance apps, and how prevalent are those categories?*

A. Manual Coding Methods

We used a manual coding process to iteratively develop and refine a codebook of app categories. For each of the 1,000 highest-ranked apps, we presented coders with (1) the app title and ID, (2) a link to a Google query for a marketplace description of each app, and (3) additional metadata for each app (e.g., installation counts, permissions, genre, etc.).

Our team consisted of four coders. We began by randomly choosing a set of 25 apps that all team members coded independently. The guidelines were to (1) assign each app one and only one code, and (2) assign codes using a two-level hierarchy of categories and sub-categories that were developed in the process of coding (e.g., *Surveillance - Location*). When no sub-category was appropriate, apps were assigned the most relevant top-level category (e.g., *Surveillance - Misc*).

After independently coding the first round of 25 apps, the group met to establish consensus and converge on appropriate code names. The results of the team's discussion were captured in a codebook that was refined in subsequent rounds of coding. We proceeded in this fashion for 4 rounds of 25, 25, 25, and 35 apps, jointly coding 110 apps. Having found that the codebook had largely stabilized after two rounds, we measured inter-coder agreement over the last 60 apps coded by the whole group. Fleiss' kappa statistic [33] indicated the coders' agreement was 0.77 when assigning apps to high-level categories, and 0.75 when assigning apps to sub-categories, indicating substantial agreement in both cases.

The remaining 890 apps were split evenly among the four coders. We took multiple precautions to ensure that coding consistency on the remaining apps would be at least as high as that attained on the 60 apps on which we measured agreement. Team members assigned a code of "other-discuss" for any app that did not fit into any category, and tagged all apps they were uncertain about as "unsure", providing explanatory comments about such apps. All apps tagged as "other-discuss" or "unsure" were reviewed by a second coder. In addition, all apps that fit into a high-level category and in a miscellaneous sub-category were reviewed to identify any trends that might only become apparent once all 1,000 apps had been reviewed. All coding modifications that resulted from this review process were discussed by the team to ensure agreement.

B. Results of Manually Coding Apps

Our algorithm captures both first-order correlation between apps that are highly likely to directly appear on devices on which our seed set of overt surveillance apps are installed, as well as apps that indirectly but strongly correlate with the seed set. The coding process revealed remarkably few apps that are not part of a clear trend; even among apps that have no obvious abusive use cases. All apps mentioned by their title here and elsewhere in the paper are listed under the code to which they pertain in Appendix C's Table IX.

The final codebook consisted of 10 high-level categories (e.g., *Surveillance*, *Harassment*, *Spoof*) and 50 sub-categories (e.g., *Surveillance - Location*, *Harassment - Social Media*, and *Spoof - SMS*). Figure 4 shows apps assigned to sub-categories, with the legend indicating the counts in parenthesis for the corresponding high-level categories. The three most prevalent sub-categories are all part of the *Surveillance* high-level category: *Surveillance - Social Media*, *Surveillance - Location*, and *Surveillance - Thorough*.

The rest of this section summarizes categories that suggest apps are used to facilitate interpersonal attacks, categories that suggest apps are used to defend against such attacks, and categories without an immediate abusive or defensive purpose. A comprehensive description of every code category, sub-category, and examples is provided in Appendix C.

Characterizing potentially abusive apps. The largest category of potentially abusive apps that we coded was *Surveillance*, which is unsurprising given that the seed set we selected consisted of surveillance apps. Apps in this category include those that (1) both covertly and overtly track someone's location, (2) record phone call audio, call metadata and call logs, (3) forward or snoop on SMS messages, (4) continuously surveil social media accounts (mostly WhatsApp and Facebook), (5) turn on the phone's camera and microphone and forward a stream to a remote device, and (6) apps that record, stream, and/or take a snapshot of a device's screen. Although CreepRank's discovery of so many surveillance apps will clearly be useful in terms of warning users about such apps or recommending that they be blocked from the app store, the nature of such surveillance apps has also been the focus of prior work [5] and thus we focus this discussion on other categories of apps, relegating the details of our surveillance-app findings to Appendix C.

We found 115 apps that enabled a variety of ways to *spoof* information, including faking images, call logs, web content, SMSs, WhatsApp messages, voice, and more. We coded 41 of these apps as *Spoof - Burner Phone* because they support the ability to make anonymous calls or SMS messages, with many explicitly advertising as useful for evading call blocking. Even more concerning and unambiguously malicious are apps that enable impersonation. Many such apps enable abusers to bait victims into a compromising response, sometimes allowing entire conversations of messages to be faked. Developers recommend their apps for putting words into the mouths of unsuspecting victims, as in the case of the "Spoof Text Message" app (see Figure 1), whose YouTube trailer[2] says, *"Don't like you buddy's girlfriend? Well, break them up! Just send a fake text message!"*. Further scrutiny of SMS spoofing apps and their malicious use cases is provided in Section VII-B.

We used *Harassment* codes to categorize apps that could be used to harass people in ways other than the mechanisms captured under surveillance, spoofing, control, and information-extraction codes (discussed elsewhere). One unexpected and prevalent type of app in this category were *fake* surveillance apps, usually marketed as prank apps, that are typically designed to be installed on a prankster's phone and briefly shown to a victim as the app simulates hacking the victim's device or accounts. Anecdotal evidence that fake-surveillance apps can cause real stress is provided by the following user review for *"Other Number Location Tracker"*, which was on the Google Play store as of June 1, 2019 and subsequently removed after we reported it to Google:

[2]https://www.youtube.com/watch?v=3MB1dVpSuRk

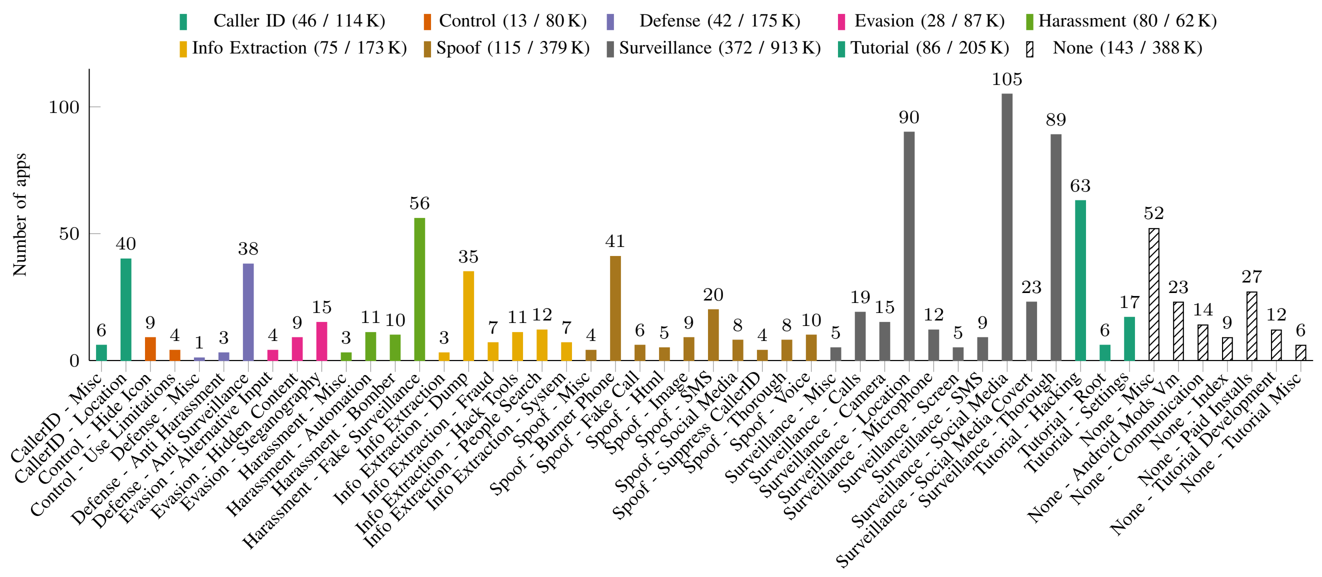


Fig. 4: Count of top 1,000 apps in each of the 50 sub-categories of our codebook. Legend shows high-level app categories and for each, the number of apps in the category and the number of app installations for that category.

> *"You say this is a joke … there is absolutely nothing funny about me looking up the number of my ex abuser who i have a restraining order against and it showing me he is 1 block away from my home. So i freak out panicking and call the cops and show them the location on my phone and they search the area and cant find him. They come back … and then … click on it just to see it is a f*cking joke!! … i hope they get shut down."*

Another concerning set of apps that we coded as *Harassment - Bomber* enable users to send high volumes of texts, calls, emails, posts, etc., to a victim. For many of these apps, such as *"Message Bomber -send 5000+ sms"*, it is difficult to envision a non-malicious use case. Section VII-B examines these apps and the context in which they are used.

We coded 86 apps as relevant *tutorials*, most of which provide *hacking* tips. Manual inspection confirmed that hacking tutorials recommend many of the attack apps that we coded (see Section VII-B), in addition to generic hacking tips, hacking term glossaries, and forums. In addition, several apps either provided tutorials for *rooting* phones, or actually rooted them, which is a vital step that enables many of the interpersonal attack apps we found.

Many of the 74 apps coded as *Information Extraction* are similar to surveillance apps in that they extract device and personal information, but not on an ongoing basis. Instead, many of these apps perform one-time dumps of content (e.g., dumping and decrypting WhatsApp databases, extracting forensic information, hidden or encrypted content caches, call logs, social media data, location history, deleted SMS messages, etc). We also found apps that directly provide hacking tools (e.g., pen-testing apps), as well as a cluster of apps that seemed most useful for *fraud*, particularly related to credit cards, which included card-number revealers, detail finders, validators, and generators. Two concerning apps are "Bank Card Validator" and "Credit Card Revealer", both of which regularly appear alongside an app that generates fake ID card images (coded as *Spoof - Image*).

A few app categories seemed useful for both attackers and victims. For example, a cluster of apps selectively hide content or are designed around privacy-focused messaging platforms, which we coded as *Evasion - Hidden-Content*. Most of these apps selectively hide images, WhatsApp content, contacts, communications, etc. They often appear alongside attack apps and are possibly used by surveillants to hide their activities from victims. In many cases these apps either hide their icons or pose as an unsuspecting app, as in the case of the "Smart Hide Calculator". We also discovered general purpose *Control - Hide-Icon* apps that hide the presence of other apps (see Section VII-B for more analysis of these apps).

Finally, we note that the above discussion of attack apps is intended to describe illustrative categories of attack apps and examples that came up in our analysis. Appendix C provides a description of every code sub-category.

Characterizing potentially defensive apps. Our coding revealed clear signs of victims protecting themselves and/or finding ways to evade restrictions imposed upon them. Although many apps assigned to other categories could plausibly have utility to both attackers and victims, we only coded apps under *defense* sub-categories when they seem to be exclusively designed to defend against surveillance or other attacks.

The most prevalent category of defense apps we discovered contained 38 *anti-surveillance* apps that prevent, block, or detect surveillance that may be conducted remotely or through physical proximity. These apps use a wide range of anti-surveillance mechanisms that includes access control, counter-surveillance of failed login attempts, and shoulder-surfing defense. For example, "Incoming Call Lock - Protector" is an access-control app that password protects incoming phone calls so that they cannot be answered by an attacker. As another example, "Oops! AppLock" enables access codes that lock the phone with no UI indications that the phone is locked, giving the impression that the phone is frozen in an open state. The unlocking mechanism is sometimes covert and subtle, such as a specific pattern of key volume presses.

A smaller category of defense apps seems to be primarily useful for victims experiencing SMS or call bombing, and remotely triggered alarms. Two such apps provided the ability to easily and temporarily disable system volume or vibrations during set times. Finally, "Hidden Apps" is a unique defensive app that reveals the presence of undesirable apps whose icons have been hidden, such as covert surveillance apps.

Characterizing apps coded as "None". Among CreepRank's top 1,000 apps are 143 that are indicative of creepware users and victims but that do not directly relate to attack or defense. Most of these apps rank towards the bottom of the top 1,000, with only 2 in the top 200. Among these, 23 apps implement *Android modifications or virtual machines*, which appeal to the hacker community and to anti-virus testers. We also observed 18 *tutorial* apps, mostly pertaining to Android modification and development, but also to catching cheating love interests. 14 *communication* apps provide group chat functionality for social-media platforms, platforms for local dating, or appear to promise free burner-phone capabilities.Finally, we found 9 *index* apps and 27 *pay-per-install* (PPI) apps, which link to many apps and incentivize users to install them. The *index* apps either directly recommend other apps, or index deals and coupons offered by other apps. The business model of PPI apps is to charge app developers who wish to artificially inflate the install counts of their apps, and then incentivize PPI app users to install these apps. Among the remaining 52 *miscellaneous* apps are several trends including money-making, social media, dating, and accessibility.

VI. Understanding CreepRank's Efficacy

The prior section highlights the wide variety of interpersonal attack and (in a few cases) defense apps identified by CreepRank's exploratory algorithm. We now discuss in more detail why CreepRank was able to find these apps by examining two questions: (1) *Does CreepRank outperform alternative algorithms such Random Walk with Restart and the MLE-based or first-order MAP approach (described in Section IV)?* and (2) *Why did some irrelevant apps show up in CreepRank's results?* In subsequent sections we further highlight CreepRank's efficacy by using it to facilitate a deeper measurement study of the creepware ecosystem.

A. CreepRank versus Alternative Algorithms

CreepRank is a single-class semi-supervised exploratory algorithm based on the principle of GBA. It differs from most malware analysis algorithms in that it does not use any descriptive features that would constrain the nature of the

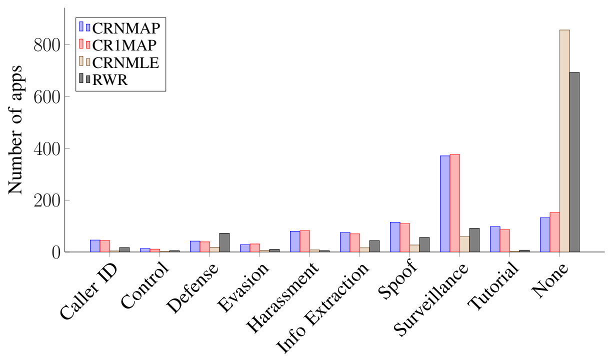


Fig. 5: Category counts for CreepRank (CRNMAP), RWR, and CreepRank variants CR1MAP and CRNMLE.

creepware apps it discovers. We compare CreepRank to Random Walk with Restart (RWR) [11], [12], another exploratory GBA algorithm that is applicable to our setting due to its use of a single class of labeled examples. The two main elements of CreepRank are its MAP estimates based on a data-driven prior belief about the scarcity of creepware apps (CRNMAP), and its ability to measure nth-order correlations between app installations. To understand which of these contributes most to CreepRank, we compare to an iterative version of CreepRank that uses maximum likelihood estimation (CRNMLE) and to a first-order correlation using maximum a posteriori probability estimation (CR1MAP) (see Section IV-C).

For the purposes of comparing these algorithms, we treat the apps coded under *None* categories as false positives, and all other apps as true positive creepware. We measure algorithmic quality based on the percentage of creepware apps in each algorithm's top 1,000 rankings. Two authors coded the top 1,000 apps produced by each algorithm, discussing possible changes to the codebook as they went, but ultimately finding that all trends were already captured by our existing codebook (Section V). Our coders achieved high inter-rater reliability over creepware categories with Cohen's kappa equal to 0.87.

A histogram of app categories in the top 1,000 results of each algorithm is shown in Figure 5. CRNMLE and RWR have the most *None* apps in their top 1,000. On the 2017 data, the top 1,000 produced by CR1MAP and CRNMAP differ by only 67 apps, yet 25 of CR1MAP's 67 are *None* apps, compared to only 5 of CRNMAP's. This suggests that for large datasets such as ours, running CreepRank iteratively until convergence yields a modest improvement in the rankings.

More important to CreepRank is the use of MAP estimation, as seen in the comparison between CRNMLE and CRNMAP. CRNMLE gave high scores to many rare apps that co-occur with creepware due to random chance, resulting in 857 *None* apps in its top 1,000 rankings, 853 of which were observed fewer than 10 times. While RWR performs moderately better than CRNMLE, it too is insufficiently skeptical of rare apps, resulting in 693 *None* apps in its top 1,000 rankings.

To see if different algorithms detected qualitatively different creepware, we examined the 307 creepware apps detected by RWR, of which 223 are not in CRNMAP's top 1,000. These FN's were typical creepware apps that fit cleanly within existing code categories, among which were 62 defensive anti-surveillance apps, 2x more detections than any other app type found by RWR. Meanwhile, CRNMLE detected 143 creepware apps, of which 122 were FN's for CRNMAP. These too were typical creepware apps, but of low prevalence. Recall that our MAP estimate deliberately sacrifices its ability to detect rare creepware apps so as to avoid CRNMLE's propensity for FP detections, which seems sensible given that rare apps affect fewer people than prevalent apps.

B. Analysis of False Positives

CreepRank's top 1,000 apps include 143 non-creepware apps that we categorized as *None*. We identified three causes for their appearance in CreepRank's top 1,000 rankings. First, the presence of *None* apps that are routinely co-installed with creepware to which they bear similarities is more or less unavoidable. For example, 18 apps were tutorials on tech and software development, which were often installed alongside hacking and creepware-focused tutorials. Another 14 other communication apps either bear similarities to burner-phone apps or provide private communications services. Among *miscellaneous* apps, 26 are similar to existing creepware apps, while the other 26 are more random, whose presence is explained by other reasons.

Second, 27 *pay-per-install* (PPI) and 9 *index* apps act as hubs [34] in the app store and would therefore be highly ranked by nearly any graph-propagation algorithm. To assess their impact on the rankings, we dropped all devices with any of the 27 PPI apps and re-ran CreepRank. The result was that 47 apps (and the 27 PPI apps) dropped from the rankings, 23 of which were *None* apps. Eight of the dropped *None* apps were money-making apps similar to PPI apps, and we conjecture that other dropped apps were advertising through PPI apps.

Finally, 23 apps create VMs or modify/emulate Android, which impact the rankings by introducing devices used for AV-testing and other atypical purposes. Through experiments described in Section VII-D, we found that eliminating 8 Android-mod apps indicative of AV-testing results in the disappearance of 15 additional *None - Android Mods VM* apps and other *None* apps. We also experimented with eliminating both PPI and AV-test apps prior to running CreepRank. This drops both FP *None* apps and TP creepware from the top 1,000, 64% of which are creepware. These lost TP's are replaced by apps that are 85% creepware, which would have improved the rankings while making them more representative of normal devices.

VII. Making Sense of the Creepware Ecosystem

The investigations described thus far uncovered a larger than expected ecosystem of creepware apps that includes many varieties of abuse apps of which we were previously unaware. Here we perform a sequence of small analyses to try to better understand this ecosystem. First, we use the context in which apps are installed to infer the most probable creepware-relevant use of apps whose intent was ambiguous or unclear (Section VII-A). Next, we seed CreepRank with various seed sets to examine the extent and character of interesting sub-categories of the creepware ecosystem (Section VII-B). We contrast profiles of attacker and victim devices in Section VII-C, and conclude this section with an investigation into the role that Norton's security app seems to play with respect to creepware (Section VII-D). Finally, we look for changes in creepware trends over time by analyzing a more recent year of data, in Section VII-E.

A. Potential Use Cases of Creepware

While coding, we hypothesized about how various types of creepware might be used. Although we have no data that directly measures usage, app installation patterns yield circumstantial evidence about how people might intend to use an app. For each category of creepware apps, we examined the context in which individual apps pertaining to the category appear. To this end, for each pair of creepware apps a and b that we coded, we calculated the *pointwise mutual information* (PMI) [35] measure, which represents the amount of information that the existence of app a has on the appearance of app b on the same device. More precisely, $\mathrm{pmi}(a;b) = \log \frac{p(a,b)}{p(a)p(b)}$ where p is the probability function. For apps in each category and those that were not confidently coded, we examined the apps that had the highest PMI values with respect to that app. To remove noise we excluded PMI values for apps that co-occurred once.

There were several instances in which our initial hypotheses about the purposes of individual apps were shown to be incorrect. In some instances, coders had envisioned a malicious use for an app that was not observed in practice. More often, we discovered unsuspected malicious uses. We now describe several examples of apps that we either re-categorized as a result of their PMI scores (these are correctly reflected in Figure 4) or that confirmed our hypotheses (see Appendix B for additional PMI data and Table IX for details of these apps):

The "Lodefast Check Cashing App" allows users to cash checks without visiting a bank. It has high PMI values with the "Card Details Finder", "Bin Checker", and "Bank Card Validator" apps, indicating that the app is likely used for fraud by some users despite good intentions by its developers.

"SMS Retaliator" seems useful for both attack and defense. We initially coded it as an anti-harassment tool because of its SMS blocking features, but PMI values indicate that it is typically used alongside message-bombing and attack apps. We saw no signs of it appearing alongside victim-side apps.

The "Unseen - No Last Seen" app is the most prevalent app for covert access to social media. This app co-occurs primarily with other covert access apps, but also with fake surveillance apps, suggesting that it is sometimes used by attackers.

The "Edit Website" app is one of several that enable users to make temporary website edits that persist until the browser is refreshed. This app provides a WYSIWIG editor for websites and is routinely installed alongside with users of spoofing, surveillance, and fake surveillance apps. Its description states that "The obvious use of this application would be to prank friends by changing headlines of news articles or paragraphs."

Apps with similar functionality that advertise for web development seem not to be used for attacks.

Finally, correlation data shows that many apps that purport to be intended for child online safety have highest PMI with apps that are unambiguously intended for intimate partner surveillance. It is unsurprising that the "Family Locator for Android" app appears alongside abuse apps, as its previous title was "GirlFriend Cell Tracker." "Cell Tracker", on the other hand, is the most prevalent app with thorough surveillance capabilities in the top 1,000 list and its marketing focuses on child safety. Although it does seem likely to be used in this way, it also correlates strongly with "Cheating Spouse", "Where the hell are you?", and "Boyfriend Tracker Free," none of which seem indicative of use on a child's phone.

B. Finding More Creepware with Alternate Seed Sets

CreepRank can also be used to surface other classes of apps. We now describe how we further explored the creepware ecosystem by running CreepRank with different app seed sets.

Seeding with Harassment - Bomber Apps. We selected the 7 bombing apps that had been most confidently coded as being entirely designed for harassment. We ran CreepRank using these apps as the seed set and coded the top 50 results (see Table IIIa). We discovered 15 more bomber apps in the top 50. We found 26 more bombers in the top 1,000 by examining the 49 apps with the following search terms in their title or app ID: SMS, bomb, dial, blast, spam, empty, blank.

Users that install bomber apps are also likely to install apps that auto-like or auto-comment on social media, presumably to bomb and harass. Interestingly, nearly all auto-liking apps do not appear to deliver on their promise, self-identifying as "pranks", with the notable exception of "404liker", which is often installed alongside malware. We found that several apps coded under *Evasion - Steganography*, because they could help abusers evade censoring, are typically co-installed with bombers. These apps create huge strings of text or emojis out of short messages or images that are sent repeatedly by bombers to amplify the impact of their attacks. These bombing attacks would be costly for victims that do not have unlimited SMS messaging. Other apps in the top 50 are in unrelated creepware categories, except perhaps for "SMS-encryption", which might be used for large string generation.

Seeding with Spoof SMS apps. To better understand how SMS Spoofing apps are used, we seeded CreepRank with 18 *Spoof - SMS* apps and coded their intent and that of the top 50 apps (see Table IIIb). Among these 68 apps, we found 32 that enable impersonation. Pernicious use of these apps, such as to damage a victim's relationships, is directly suggested in marketing materials for some of these apps (see Figure 1). Such apps could also be used to elicit compromising responses from intimate partners that are suspected of infidelity, similar to the attack suggested by the tutorial app in Figure 6b. Of particular interest are eight impersonation apps that enable entire conversations to be falsified, which seem to be mostly about constructing false evidence, such as the "Sending Fake SMS app", which markets itself to unfaithful intimate partners for falsifying alibis. Several others are intended for installation on a victim's phone, where mimicked SMS, Facebook, or WhatsApp notifications sent by the abuser can cause the victim to open the spoofing app thinking that they have received a genuine message from whomever the abuser chose to impersonate. These apps have clear parallels to phishing attacks but are under-studied.

PMI values indicate that the 15 anonymity-focused apps are used by abusers more than victims, possibly to send anonymous messages that are difficult to block. "SMS Receive" and similar apps enable users to receive messages at shared anonymized numbers, such as for 2-factor authentication notifications, and in conjunction with apps that provide burner-phone and temporary email services. Rounding out the top 50 are 19 attack apps (mostly surveillance) and 2 defensive apps.

Seeding with Control-Hide Icon apps. Table IIIc shows the top 50 results when we seed CreepRank with nine icon-hiding apps. The top 50 includes 12 app-hiders, three of which camouflage other apps by changing their icons or metadata, while the rest hide app icons from the user interface. Several apps hide their own icons, while others camouflage themselves by posing as a calculator, currency exchanger, or flashlight. Three of the hidden apps can only be opened by calling a fake phone number, while most other hidden-content apps require some sort of passcode. Users of app-hiders frequently install apps that hide content, many of which provide dual public and secret channels for content and/or communication. Also noteworthy is "Hidden Apps", a defensive app that reveals the presence of hidden apps. The top 50 contained 9 additional defensive apps that provide access control for some combination of the device itself, its apps, and incoming phone calls. Eight other attack apps round out the top 50 results.

Hacking Tutorials. For further confirmation of our hypotheses about how creepware apps are used, we turned to hacking tutorial apps. We installed the hacking tutorials that were prominent in CreepRank's results or had high PMI scores with abusive apps. We now describe three such tutorials.

"SpyBoy" was notable for its high PMI scores with attack apps across many categories. It confirmed many of our hypotheses by describing interpersonal attacks that cover a remarkably large fraction of the creepware categories we identified, including: email, HTML, SMS, and caller spoofing, use of hack tools, remote control of devices, secret settings for attack and defense, steganography, imposing WiFi use limitations on other devices, and several categories of surveillance.

"Top Spy Apps" gives a ranked list of interpersonal surveillance apps in general, and intimate partner surveillance apps in particular (see Figure 6a). Each app has a page where it is described and extolled for it's best spying features.

"Cheating spouse tracker" includes vivid descriptions of how to entrap a cheating spouse, recommending specific surveillance apps (see Figure 6b).

Count	Code
15	Bomber
14	Auto-Liking
10	Evasion-Steganograpy
6	Info Extraction
1	SMS Encryption
1	Harassment
1	Spoof
1	Surveillance

(a) Seed set of 7 Bomber apps

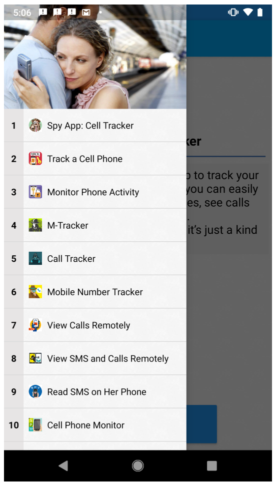
Count	Code
13 (+11 seed set)	Impersonate Sender
5 (+3 seed set)	Impersonate Conversation
11 (+4 seed set)	Anonymity
13	Surveillance and Dumping
4	Fake Surveillance
2	Spoof - Call Logs
1	SMS Blacklist
1	Evasion - Hidden Content

(b) Seed set of 18 SMS Spoofing apps

Count	Code
20	Hidden Content
12	Hide Apps
9	Access Control
6	Surveillance
1	Hidden App Finder
1	Info Extraction
1	Fake Surveillance

(c) Seed set of 9 App Hiders

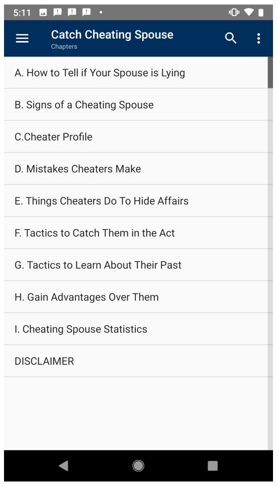
Table III: Coding results of the top 50 apps produced by CreepRank on 2017 install data when seeded with apps of different categories. In the case of SMS spoofing apps, we re-coded the apps to capture their nuanced functionality.


5:06

1 Spy App: Cell Tracker
2 Track a Cell Phone
3 Monitor Phone Activity
4 M-Tracker
5 Call Tracker
6 Mobile Number Tracker
7 View Calls Remotely
8 View SMS and Calls Remotely
9 Read SMS on Her Phone
10 Cell Phone Monitor

ker
o to track your
you can easily
es, see calls
it's just a kind

(a)


5:11

Catch Cheating Spouse
Chapters

A. How to Tell if Your Spouse is Lying
B. Signs of a Cheating Spouse
C.Cheater Profile
D. Mistakes Cheaters Make
E. Things Cheaters Do To Hide Affairs
F. Tactics to Catch Them in the Act
G. Tactics to Learn About Their Past
H. Gain Advantages Over Them
I. Cheating Spouse Statistics
DISCLAIMER

(b)

Fig. 6: (a) "Top Spy Apps" lists spyware apps and their uses for interpersonal surveillance. (b) "Cheating spouse tracker" includes guides recommending specific surveillance apps.

Y = *Harassment – Bombers*		Y = *Surveillance – Location*	
App Category X	Δ_{XY}	**App Category** X	Δ_{XY}
Evasion – Steganography	5.79	CallerID – Location	1.52
Harassment – Automation	5.59	Harassment – Fake Surveillance	1.27
Spoof – HTML	3.19	Surveillance – Thorough	0.62
Spoof – Misc	2.71	Surveillance – SMS	0.38
Defense – Anti-harassment	2.59	Defense – Misc	0.17
Surveillance – Calls	-0.11	None – Android Mods VM	-0.63
Surveillance – Camera	-0.13	Evasion – Steganography	-0.63
CallerID – Misc	-0.15	Harassment – Misc	-0.68
Surveillance – Location	-0.58	Tutorial – Root	-0.75
CallerID – Location	-0.60	Spoof – Misc	-0.75

Table IV: Relative difference Δ_{XY} between probabilities that a device has an app from category X given that it has an app from category Y = *Harassment – Bombers* or Y = *Surveillance – Location* vs. it has an app from category X.

C. Characterizing Devices via Creepware

To better understand the nature of devices with creepware installed, we analyze correlations between different categories of apps co-installed on devices. For example, we hypothesize that certain apps are typically installed on devices being used by an abuser, while other apps are primarily installed on victim devices. Let Y be a category of apps conjectured to be indicative of a device's role. We focus on Y being *Harassment – Bombers* (likely installed on the device of abusers) or *Surveillance – Location* apps (likely installed on the device of a victim). The tables in Table IV show the five highest and lowest app categories X for the two Y categories, where the ranking for category X is calculated as the relative difference $\Delta_{XY} = (\Pr(X|Y) - \Pr(X))/\Pr(X)$ where $\Pr(X)$ is the probability of observing at least one app with category X on a device, and $\Pr(X|Y)$ is the probability of observing at least one app with category X on a device given that the device has at least one app with category Y.

As can be seen, harassment apps tend to be installed on the same device as other harassment apps: the top four apps for Y being *Harassment – Bombers* are all categories of apps useful for sending harassing messages. For Y being *Surveillance – Location* the situation is almost exactly reversed, with the top four app categories being spyware related. This suggests that, in some cases, it may be possible to characterize devices as attacker-owned or victim-owned based on the types of apps installed. Whether such predictions can be made accurate or useful remains an open question.

D. Role of the Norton Mobile Security App

By obtaining our dataset from a security vendor, we only have data from devices on which the vendor's app is installed. We wanted to investigate if the Norton app was most often used preventatively or for post-infection cleanup. We identified 172 K devices on which the Norton app was installed alongside one or more of the 107 thorough surveillance apps we identified (including apps in CreepRank's seed set). We then dropped about 8 K potentially anomalous devices that had more than 1 K apps installed in any one year. In 22 K of the remaining 164 K devices, the Norton app was installed after a surveillance app, suggesting post-infection cleanup. For the rest of the devices, the security app was installed before the surveillance app, suggesting it is being used preventatively.

This leads us to ask why an attacker would install a security app on their device? A possible reason is that attackers are frequently engaging in risky behaviors, such as installing questionable or off-store apps and rooting devices. Thus, they may use the security app to guard against possible compromise.

No Device Filtering		AV Device Filtering	
Count	Code	Count	Code
19	Malware	4	Malware
1	Not Found	3	Not Found
		11	Surveillance
		1	Spoof Social Media
		1	Anti-Surveillance

Table V: Coding of top 20 apps for which we lacked marketplace data, with and without AV-test device filtering

AV-Testing Devices and Offstore Apps: To identify devices used in AV-testing, we examined apps coded as *None - Android Mods / VM* that may be used for AV testing. Seven of these apps (e.g., apps that emulate Linux or Chromium or enable software development) seemed unlikely to be installed on typical user devices. We also noticed that the Appium Mobile App Automation toolkit [36] often appeared on devices with malware and other Android-modification apps, which suggests its use in AV testing. Thus, we added Appium to the other seven AV-testing apps, removed devices containing any of these eight apps from the data, and re-ran CreepRank. The effects on the overall rankings of apps that appeared in Norton's marketplace data were modest, except for a few additional Android-modification apps that dropped precipitously in the rankings, likely because they were also used in testing.

We expected that filtering out AV-test devices would have a large impact on CreepRank's rankings of malware apps. To analyze this, we compared the top 20 ranked apps, both with and without filtering, for which Norton did not have marketplace data (see Table V). We found that filtering devices with apps indicative of AV testing has a dramatic impact on the rankings, with no overlapping apps between its top 20 list and the unfiltered top 20 list. The unfiltered top 20 list consists primarily of malware apps on devices where Appium automation apps appear. By contrast, CreepRank's top 20 list for filtered devices consists of apps that appear to have existed on the Google Play store at one time, but probably only briefly, as few of the sites that scrape the Google Play store have records of these apps. Most were surveillance apps, a few were malware, and there was one defense and one spoofing app. We could not find any useful information about 3 apps.

E. Creepware over Time

We now examine how creepware evolves over time by running CreepRank on the 2018/19 dataset (spanning May 1st 2018 to May 1st 2019) and comparing the results to the 2017 dataset. As noted above, CreepRank tends to perform better as the number of devices infected by its seed set increases. However, the 18 surveillance apps used as the original seed set on the 2017 data had declined in popularity by 2018/19. To compensate, we added another 32 thorough surveillance apps that CreepRank identified in the 2017 data, selecting apps that were prevalent in 2018/19. This resulted in a seed set of 50 apps installed on 32,719 devices in 2018/19, compared to 18 apps installed on 35,811 devices in 2017.

Category Counts			Largest Sub-Category Change			
Category	2017	2018/19	Sub-Category	2017	2018/19	Δ
CallerID	46	11	Location	40	11	29
Control	13	1	Hide Icon	9	0	9
Defense	42	**58**	**Anti-Surveil**	38	**51**	**13**
Evasion	28	16	Steganography	15	0	15
Harassment	80	39	Fake-Surveil	56	24	32
Info extraction	75	**164**	**Hack-Tools**	11	**70**	**59**
Spoof	115	54	Burner-Phone	41	15	26
Surveillance	372	**445**	**Social-Media**	105	**179**	**74**
Tutorial	86	72	Hacking	63	44	19
None	143	140	Pay-Per-Install	27	1	26

Table VI: Count of app categories in CreepRank's top 1,000 for 2017 and 2018/19 data and, within each category, the sub-category with the greatest change (growth in bold).

We ran CreepRank on the 2018/19 data and following the same procedures as before, three authors coded the 2018/19 top 1,000 ranked apps. We then reviewed the top 1,000 to identify trends and determine if any new categories of creepware had emerged, but found that the existing codebook covered all common cases. Many 2017 apps fell out of use in 2018/19 and the two top 1,000 lists overlap by only 110 apps, suggesting there are significant changes to the creepware ecosystem over time.

Table VI shows the total number of apps in each category across 2017 and 2018/19, as well as, for each category, the sub-category with the largest change between the time periods. From the table, it is clear that the privacy of creepware victims is still under assault. The information extraction category more than doubled, with hacking tools the largest area of growth. The number of surveillance apps also grew substantially, with increases in social media, microphone, SMS, and thorough surveillance apps more than making up for a nearly 50% drop in location surveillance apps. Interestingly, we did not find many new spoofing apps, although 21 apps from the 2017 data were still active and among the most popular apps, by installation count, in 2018/19.

On a more positive note, although the number of social media surveillance apps grew in 2018/19, our analysis of these new spying apps suggests that new security precautions by WhatsApp in particular have curtailed access to message content, leaving these apps to report on usage statistics and little else. We also noticed an increase in the fraction of surveillance apps that are recommended for child safety use relative to intimate partner surveillance, which could indicate a change in how developers are advertising their surveillance apps, perhaps in response to Google's policy and enforcement changes as a consequence of recent studies [5].

VIII. Discussion

Practical impact. The analyses described in previous sections suggest that CreepRank is a valuable tool for discovering and making sense of a broad range of apps used in interpersonal attacks and, to a lesser extent, defense. These findings have already proven practically useful. Thus far, Norton has begun to scan and warn customers about CreepRank-identified apps that were verified as creepware by our manual coding. These apps are also now flagged as potentially dangerous by the IPV Spyware Discovery tool, which is used in Cornell Tech's computer security clinic for IPV survivors [9], [10].

We also went through a responsible disclosure process with Google to report 1,095 apps we discovered that may have been on the Google Play store in violation of their policies. Google Play provides policies designed to prevent abusive apps like creepware. Its Potentially Harmful Applications policy [37] focuses mostly on malware prevention. More related is the "Privacy, Security, and Deception" portion of Google's Developer Policy Center [38], whose sub-policies on "Device and Network Abuse", "Malicious Behavior", and "Deceptive Behavior" contain many rules that prohibit creepware functionality. Particularly prohibited are spoofing and fake-surveillance apps that "attempt to deceive users or enable dishonest behavior"; fraud-based fake-ID apps that "generate or facilitate the generation of ID cards"; hacking tools and tutorials that "facilitate or provide instructions on how to hack services, software or hardware, or circumvent security protections"; and surveillance and commercial spyware apps. The policy also states that "Any claim that an app is a 'prank', 'for entertainment purposes' (or other synonym) does not exempt an app from application of our policies." Google ultimately determined that 813 of the 1,095 creepware apps we reported violate their policies, and those have been removed.

The creepware problem moving forward. CreepRank enabled the first measurement study of the broad creepware ecosystem, and this measurement study has, in turn, already had positive practical impact by surfacing a large set of verified creepware. Our results suggest that creepware is a widespread problem and this raises a number of tricky questions about how to mitigate their harms moving forward.

Keeping creepware out of app stores will be challenging. New apps tend to rise in the place of removed apps, and developers attempt to obfuscate their app's purpose in order to evade policy enforcement. For example, recent bombing apps use the term "text repeater" and avoid direct references to bombing. While this may make these apps harder for attackers to find, it also makes it harder to enforce policy at scale.

A next step would be to create and deploy a system capable of detecting creepware in a (semi-)automated fashion. CreepRank provides a starting point and could be used as a first step to identify candidate creepware apps, manually verify them to generate labeled training data, and then use this data to train machine learning classifiers to detect surveillance, spoofing, harassment, and other pernicious app categories. Further work is needed to develop and evaluate such an approach, including gauging how often one would need to update CreepRank's output, how many labeled apps are needed, what types of features are effective to use, and more.

A particular challenge facing such an approach would be dealing with data poisoning attacks, in which attackers attempt to evade detection by, for example, gaming an app's CreepRank. This is related to the challenge of detecting emulated testing and research devices, since such emulation could be used to inject malicious co-installation patterns. As discussed in Section VII-D, we observed in our dataset some devices that could fall into this category. While we do not believe these affected our measurement study results thus far, should CreepRank or similar approaches be put to use moving forward, we may have to contend with deployment of malicious emulation or research devices that pollute data. Ancillary measures such as the detection of cloned devices may help, and we leave exploring these issues to future work.

Even with good detection capabilities, deploying detection tools raises a host of questions. In addition to screening of app stores, we would like to directly integrate creepware detection into a commercially available anti-virus software. But making creepware detection actionable for users remains a challenge. Much of the creepware we discovered are harassment apps that are installed on abuser devices, and issuing creepware notifications to abusers may not be useful. Whether and how one can craft messaging to deter interpersonal attackers are important questions for future work.

For creepware that is installed on a victim's device, questions remain regarding how and when to notify them. For instance, if the AV notifies the user immediately (as done currently), an abuser with physical access to the device might dismiss or ignore the warnings and disable the detection software. But if the detection software attempts to wait until it is more certain that the original owner has possession of the device, there are still issues of victim safety. For instance, removal of creepware could result in escalation of interpersonal attacks to physical violence in cases of IPV. This threat might be mitigated by designing notifications that attempt to take safety planning into consideration, which would require additional exploration.

IX. Conclusion

We explored the landscape of apps that are likely to be used in interpersonal attacks, called creepware. We created CreepRank, an exploratory algorithm based on the principle of guilt by association, and ran it on a dataset of billions of app installations. We discovered and explored many categories of apps that enable surveillance, harassment, impersonation, information theft, concealment, and more. Our methods and analysis of creepware are useful for app stores and anti-virus vendors seeking to improve safety for mobile device users.

Acknowledgements

The authors would like to thank our shepherd Emily Stark and the anonymous reviewers of our study for their feedback and suggestions to improve the quality of our manuscript. We acknowledge funding support under NSF award numbers 1717062, 1916096, 1916126, and gifts from Google.

References

[1] D. Freed, J. Palmer, D. Minchala, K. Levy, T. Ristenpart, and N. Dell, "Digital technologies and intimate partner violence: A qualitative analysis with multiple stakeholders," PACM: Human-Computer Interaction: Computer-Supported Cooperative Work and Social Computing (CSCW), vol. 1, no. 2, p. Article 46, 2017.

[2] ——, “A Stalker’s Paradise: How Intimate Partner Abusers Exploit Technology,” in Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI). New York, NY, USA: ACM, 2018, pp. 667:1–667:13.

[3] T. Matthews, K. O’Leary, A. Turner, M. Sleeper, J. P. Woelfer, M. Shelton, C. Manthorne, E. F. Churchill, and S. Consolvo, “Stories from survivors: Privacy & security practices when coping with intimate partner abuse,” in 2017 CHI Conference on Human Factors in Computing Systems (CHI), 2017, pp. 2189–2201.

[4] N. Sambasivan, A. Batool, N. Ahmed, T. Matthews, K. Thomas, L. S. Gaytan-Lugo, D. Nemer, E. Bursztein, E. F. Churchill, and S. Consolvo, ““They Don’t Leave Us Alone Anywhere We Go”: Gender and Digital Abuse in South Asia,” in CHI Conference on Human Factors in Computing Systems (CHI), 2019.

[5] R. Chatterjee, P. Doerfler, H. Orgad, S. Havron, J. Palmer, D. Freed, K. Levy, N. Dell, D. McCoy, and T. Ristenpart, “The spyware used in intimate partner violence,” in IEEE Symposium on Security and Privacy (S&P), 2018, pp. 441–458.

[6] Y. Ye, T. Li, S. Zhu, W. Zhuang, E. Tas, U. Gupta, and M. Abdulhayoglu, “Combining file content and file relations for cloud based malware detection,” in International Conference on Knowledge Discovery and Data Mining (KDD), 2011, pp. 222–230.

[7] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitraş, “The dropper effect: Insights into malware distribution with downloader graph analytics,” in Conference on Computer and Communications Security (CCS), 2015, pp. 1118–1129.

[8] A. Tamersoy, K. A. Roundy, and D. H. Chau, “Guilt by association: large scale malware detection by mining file-relation graphs,” in International Conference on Knowledge Discovery and Data Mining (KDD), 2014, pp. 1524–1533.

[9] S. Havron, D. Freed, R. Chatterjee, D. McCoy, N. Dell, and T. Ristenpart, “Clinical computer security for victims of intimate partner violence,” in USENIX Security Symposium, 2019, pp. 105–122.

[10] D. Freed, S. Havron, E. Tseng, A. Gallardo, R. Chatterjee, T. Ristenpart, and N. Dell, ““Is my phone hacked?” Analyzing clinical computer security interventions with survivors of intimate partner violence,” PACM: Human-Computer Interaction: Computer-Supported Cooperative Work and Social Computing (CSCW), vol. 3, pp. 202:1–202:24, 2019.

[11] L. Grady, “Random walks for image segmentation,” IEEE Transactions on Pattern Analysis & Machine Intelligence, no. 11, pp. 1768–1783, 2006.

[12] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu, “Automatic multimedia cross-modal correlation discovery,” in International Conference on Knowledge Discovery and Data Mining (KDD), 2004, pp. 653–658.

[13] SpoofBox, “Spoof text message trailer,” https://www.spoofbox.com/en/preview/spoof-text, 2019, online; accessed 18 Nov 2019.

[14] N. E. Willard, Cyberbullying and cyberthreats: Responding to the challenge of online social aggression, threats, and distress. Research press, 2007.

[15] P. K. Smith, J. Mahdavi, M. Carvalho, S. Fisher, S. Russell, and N. Tippett, “Cyberbullying: Its nature and impact in secondary school pupils,” Journal of child psychology and psychiatry, vol. 49, no. 4, pp. 376–385, 2008.

[16] B. Farinholt, M. Rezaeirad, P. Pearce, H. Dharmdasani, H. Yin, S. Le Blond, D. McCoy, and K. Levchenko, “To catch a ratter: Monitoring the behavior of amateur DarkComet RAT operators in the wild,” in IEEE Symposium on Security and Privacy (S&P), 2017, pp. 770–787.

[17] S. Le Blond, A. Uritesc, C. Gilbert, Z. L. Chua, P. Saxena, and E. Kirda, “A Look at Targeted Attacks Through the Lens of an NGO,” in USENIX Security Symposium, 2014, pp. 543–558.

[18] W. R. Marczak, J. Scott-Railton, M. Marquis-Boire, and V. Paxson, “When governments hack opponents: A look at actors and technology,” in USENIX Security Symposium, 2014, pp. 511–525.

[19] P. Kotzias, L. Bilge, and J. Caballero, “Measuring PUP prevalence and PUP distribution through pay-per-install services.” in USENIX Security Symposium, 2016, pp. 739–756.

[20] K. Thomas, J. A. E. Crespo, R. Rasti, J. M. Picod, C. Phillips, M.-A. Decoste, C. Sharp, F. Tirelo, A. Tofigh, M.-A. Courteau, M.-A. Courteau, L. Ballard, R. Shield, N. Jagpal, M. A. Rajab, P. Mavrommatis, N. Provos, E. Bursztein, and D. McCoy, “Investigating commercial pay-per-install and the distribution of unwanted software.” in USENIX Security Symposium, 2016, pp. 721–739.

[21] Y. Hu, H. Wang, L. Li, Y. Guo, G. Xu, and R. He, “Want to earn a few extra bucks? a first look at money-making apps,” in IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 2019, pp. 332–343.

[22] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, “A survey of mobile malware in the wild,” in ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM), 2011, pp. 3–14.

[23] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, “DREBIN: Effective and explainable detection of android malware in your pocket.” in Network and Distributed Systems Security Symposium (NDSS), 2014, pp. 23–26.

[24] Z. Yuan, Y. Lu, and Y. Xue, “Droiddetector: Android malware characterization and detection using deep learning,” Tsinghua Science and Technology, vol. 21, no. 1, pp. 114–123, 2016.

[25] M. Hatada and T. Mori, “Detecting and classifying Android PUAs by similarity of DNS queries,” in IEEE Annual Computer Software and Applications Conference (COMPSAC), vol. 2, July 2017, pp. 590–595.

[26] Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, “Taming information-stealing smartphone applications (on Android),” in Trust and Trustworthy Computing (TRUST), 2011, pp. 93–107.

[27] Y. Aafer, W. Du, and H. Yin, “Droidapiminer: Mining API-level features for robust malware detection in Android,” in International conference on security and privacy in communication systems (SecureComm), 2013, pp. 86–103.

[28] D. H. Chau, C. Nachenberg, J. Wilhelm, A. Wright, , and C. Faloutsos, “Polonium: Tera-scale graph mining and inference for malware detection,” in SIAM International Conference on Data Mining (SDM), 2011.

[29] J. Yoo, S. Jo, and U. Kang, “Supervised belief propagation: Scalable supervised inference on attributed networks,” in 2017 IEEE International Conference on Data Mining (ICDM), 2017, pp. 595–604.

[30] S. Seneviratne, A. Seneviratne, P. Mohapatra, and A. Mahanti, “Your installed apps reveal your gender and more!” SIGMOBILE Mobile Computing Communications Review (SIGMOBILE), vol. 18, no. 3, pp. 55–61, Jan. 2015. [Online]. Available: http://doi.acm.org/10.1145/2721896.2721908

[31] E. Malmi and I. Weber, “You are what apps you use: Demographic prediction based on user’s apps,” in International Conference on Web and Social Media (ICWSM), 2016, pp. 635–638.

[32] M. J. Breiding, M. C. Black, and G. W. Ryan, “Prevalence and risk factors of intimate partner violence in eighteen U.S. states/territories, 2005,” American Journal of Preventative Medicine, vol. 34, no. 2, pp. 112–118, 2008.

[33] J. L. Fleiss, “Measuring nominal scale agreement among many raters,” Psychological bulletin, vol. 76, no. 5, p. 378, 1971.

[34] J. M. Kleinberg, “Authoritative sources in a hyperlinked environment,” Journal of the ACM, vol. 46, no. 5, pp. 604–632, Sep 1999.

[35] K. Ward Church and P. Hanks, “Word association norms, mutual information, and lexicography,” Computational Linguistics, vol. 16, no. 1, pp. 22–29, 1990.

[36] N. Verma, Mobile Test Automation With Appium. Packt Publishing, 2017.

[37] “Google play protect - potentially harmful application (PHAs) categories,” https://developers.google.com/android/play-protect/phacategories, online; accessed 18 Nov 2019.

[38] “Google play store - developer policy center,” https://play.google.com/about/developer-content-policy/, online; accessed 18 Nov 2019.

Appendix

A. Seed Set Apps

Table VII shows the 18 covert surveillance apps that we used as the seed set for running CreepRank on 2017 Norton app installation data. These apps were identified by Chatterjee et al. [5] as covert surveillance apps that are primarily distributed outside of Android app stores such as Google Play.

B. Examples of PMI Analyses

In Section VII-A we provide examples of apps whose primary use case was unclear during coding, and for which the Pointwise Mutual Information (PMI) metric gave us valuable insights into the context in which these apps are most often

Title	Package Name
System Service	com.android.system
Wi-Fi Settings	com.wifiset.service
Data Backup	com.spy2mobile.light
Sync Service	com.android.core.monitor.debug
System Service	com.mxspy
Backup	com.spappm_mondow.alarm
SystemTask!	com.spytoapp.system
System Service	com.guest
System Services	com.topspy.system
UPreferences	com.android.preference.voice
Secure Service	com.safesecureservice
Mobile Spy	com.gpssettings.src.v65
System Service	com.ispyoo
Update service	sys.framework
com.android.devicelogs	com.android.system.devicelogs
Internet Service	com.sec.android.internet.service.ik
System Services	com.hellospy.system
System Update Service	com.ws.sc

Table VII: Seed set of surveillance apps used by CreepRank on 2017 app installation data [5].

used in practice. Here we provide more details in Table VIII which lists, for the five apps described in that section, the ten apps with the highest corresponding PMI scores. To remove noise from the PMI rankings, we exclude PMI values for apps that co-occurred with with the target app only once.

C. Description of Codebook

This appendix describes the codes developed during analysis of the top 1,000 apps produced by CreepRank on 2017 data. All counts provided here are in reference to the CreepRank 2017 top 1,000 apps. Further coding of 2018/2019 data did not require modifications to the codebook.

Caller ID - Misc: 6 apps provide caller ID functionality that did not fit into other sub-categories. These apps often enable call blocking or claim to reveal private phone numbers.

Caller ID - Location: 40 apps bundle caller-id functionality with location tracking of placed calls, usually claiming cell-tower information as the source of location information (when the source is specified at all). Many of these apps claim to be able to use cell tower information to determine the location of incoming calls. Co-occurrence data suggests these apps are used by attackers and victims alike.

Control – Hide Icon: 9 apps hide the icons of other apps from the home screen or app launcher screen (usually both), rendering their presence sufficiently covert as to be unlikely to be noticed and removed.

Control – Use Limitations: 4 apps enable the user to interrupt the internet access of other devices on the WiFi network, typically by staging ARP-spoofing network attacks. The “NetCut” app is one of many such apps.

Defense – Misc: “Hidden Apps” is the sole miscellaneous defense app among the 2017 top 1,000 apps. It is a hidden app-icon revealer used to counter *Control - Hide Icon* apps.

Defense – Anti Harassment: We found 3 apps that appear to be used to mitigate call and SMS bombing. These apps are frequently co-installed with other bombing-defense apps, such as apps with caller-id and caller blacklisting functionality. Two of these apps facilitate muting the device’s system volume and/or vibration alerts during set times or for a period of time, while another app blacklists SMS senders.

Defense – Anti Surveillance: 38 apps counter surveillance attempts through a variety of means, including access control for incoming phone calls and individual apps, counter-surveillance, and shoulder-surfing defense. Specific examples are provided in Section V-B.

Evasion – Alternative Input: 4 apps provide accessibility-focused mechanisms for enabling user input, such as control of a mouse-style pointer based on the tracking of face or eye movement. These apps are routinely installed alongside surveillance apps that monitor the device and enable it to be controlled remotely.

Evasion – Hidden Content: Most apps in this category selectively hide content including images, WhatsApp content, contacts, and communications. We found 9 apps when coding 2017 data, and many more when seeding CreepRank with Control - Hide Icon apps. Many of these apps either hide their icons or pose as an unsuspecting app, as in the case of the “Smart Hide Calculator”. We also found several messaging platform apps designed specifically for secrecy, both through heavy use of encryption and ephemeral messaging.

Evasion – Steganography: A few apps, such as “Pixel-knot”, use steganography to hide messages in images such that a human would not perceive them, suggesting a desire for secrecy. However, most of the apps we found created messages that can be interpreted by humans but not readily interpreted by machines, such as emojifying apps and apps that convert words to images or ASCII art. We were surprised to find that most of these were being used by attackers, possibly to avoid censorship on social media platforms.

As discussed in Section VII-B, there are several apps that we coded initially as steganography apps that actually seem to be used to amplify the effect of SMS bombing attacks. These apps create ASCII art or do image-to-text conversion, creating huge strings that are sent over and over by bombing apps.

Harassment – Automation: 11 apps automate social media activity, such as auto-reply, schedule-based message senders, auto-commenting, and chatbots. These apps show varying degrees of malicious intent, typically correlating most strongly with social media surveillance apps, bombing apps, or malware. Most of these apps focus on WhatsApp.

Harassment – Bomber: 10 apps are designed to send high volumes of texts, calls, emails, social media posts, etc., though a few instead send messages that are so large as to cause a serious nuisance or cost to the victim. These apps seem to have a short shelf life on app stores, but new apps rise to take their place. Recent apps of this ilk are more likely to refer to themselves as “text repeaters” than as “bombers.”

Harassment – Fake Surveillance: The most prevalent harassment apps are the 56 apps that scare victims by giving them a false impression that they are being surveilled. Most of these self-described “prank” apps can be installed on the abuser’s device and shown briefly to the victim while the app simulates hacking of the victim’s device or accounts.

Information Extraction – Misc: 3 apps did not fit into sub-

Cell Tracker	Call Spoofer	Lodefast Check Cashing App	SMS Retaliator	Unseen No Last Seen
Mobile Phone Tracker	Fake Call (1)	Card Details Finder	AirMon	No Last Seen or Read
Cell Phone Tracker Tutorial	Spoof Call	Bin Checker	Droidbug Pentesting Forensic FREE	Private Read for FB Messenger
Mobile Phone Tracked	Phone Gangster Coupon	Bank Card Validator	PirateBox	SpyGo For Whatsa Prank
TangTracker e-Safety App	Spoof my Phone	Free People Search Peek You	Manual Hacker Gold	Invisible Chat for Facebook
Free Cell Phone Tracker	Spoof Caller	Credit Card Revealer	Bugtroid Pentesting FREE	Unseen: no seen marks
Cheating Spouse	Untraceable Calls	Credit Card Validator with CVV	Bugtroid Pentesting PRO	Last seen online hider for whatsapp
Cell Phone Tracker Number	Spoof SMSPhone	Free People Search Public Records	IPConfig	Blue tick
Mobile Tracker	Phone Id Faker	CallLog & SMS Tracker	HTTP Tools	hack and pirate face prank
GuestSpy: Mobile Tracker	Spoof SMS Sender	SpyFly	Wicap. Sniffer Demo ROOT	WhatsOn for Whatsapp
Where the hell are you?	Fake Call (2)	AWS Code Viewer	Super Download - Booster	WhatsSpy VIP! PRANK

Table VIII: For five example apps, we show the top 10 apps that co-occur at least twice, in order of descending PMI scores.

categories, including two with decryption functionality and one that captures extended screenshots of content.

Information Extraction – Dump: 32 apps perform large-scale dumps of a broad variety of content, which include WhatsApp database decryptors and dumpers, extractors of forensic information, call logs, social media contacts, location history, deleted content, hidden or encrypted content, etc.

Information Extraction – Fraud: We found 6 apps with use cases that pertain to fraud, such as credit card number revealers, details finders, validators, and generators. Two examples are the "Bank Card Validator" and the "Credit Card Revealer" app. Apps that generate fake ID card images routinely appear alongside credit card revealing apps, which strengthens the hypothesis that they the card revealers are used for fraud. The "Lodefast Check Cashing App" enables the cashing of checks without visiting a bank, and we re-classified it under fraud when PMI values revealed that it is usually installed alongside fraud apps in our dataset (see Table VIII).

Information Extraction – Hack Tools: 11 apps provide hacking tools, three of which focus on extracting passwords, while one looks passwords up in public data breach repositories. The remainder enable sniffing of wireless network traffic or provide pen-testing and attack functionality. There was a noticeable increase in the number of hacking apps in 2018/2019 data, among which sniffing apps were very prevalent.

Information Extraction – People Search: 12 apps look up personal details pertaining to individuals. Searches may be keyed off of phone number, names, email addresses, etc, frequently providing extensive personal information. The most unique app in this category is "BaeList", which advertises as a tool to catch cheaters by alerting its users if a suspected cheater's phone number has been searched for by another user.

Information Extraction – System: 7 apps extract Android system details, such as IP address, IMEI, and SIM cards. The purposes of such apps are usually left unspecified, but they are useful for Control - Use Limitations apps and for network-based surveillance tools.

None – Misc: 124 apps have no discernible utility for an attacker or victim. Most of these apps fit cleanly into sub-categories described below. Of the 55 apps that do not fit into sub-categories, around half are false positives introduced by Pay-Per-Install apps, AV-testing (some of these apps are used extensively as benign examples), and cloned devices. In a few cases, app titles suggest malicious functionality that is not delivered, such as the deceptively named "Spy Mobile" app. The remainder of these apps correlate strongly with malicious creepware, and the presence of some apps, such as the "Blue Whale Game," is alarming, as it issues a series of self-harm challenges and culminates in a suicide challenge.

None – Android and OS Mods: 21 apps modify or extend Android, such as emulating the Chromium OS, adding windowing support, etc. It is evident from the number of hacking-related apps and tutorials in the data that the hacker community makes ample use of creepware, and co-occurrence data suggests such users are likely to root their devices and experiment with OS modification. Also contributing to the presence of these apps are significant numbers of AV-testing and researcher devices, as discussed in Section VII-D.

None – Communication: 13 apps provide communication functionality, such as extending WhatsApp with group messaging capabilities, providing free SMS or phone calls, or enabling walkie-talkie functionality. Many of these apps advertise as ways to meet local singles.

None – Index: The primary purpose of 9 apps is to provide indices of items on sale or of money-making opportunities. The former primarily index online deals, though many indirectly encourage the installation of additional apps.

None – Pay Per Install (PPI): 27 apps incentivize users to install other apps on their devices, primarily by offering payments or free calling services. App developers that advertise through PPI apps are able to artificially increase the installation counts of their apps and receive fake favorable reviews.

None – Tutorial Misc: 11 tutorial apps did not fit into a strong trend, including apps that teach skills useful for hacking but that do not mention hacking explicitly (e.g., DOS CMD commands). Those that seem most benign use the word "hack" in their titles, which may have led to their being downloaded under false expectations.

None – Tutorial Development: 6 apps focus on app development, half of which provide the ability for non-technical users to create their own apps (e.g., by providing templates).

Spoofing – Misc: 4 apps that provide spoofing functionality do not fit cleanly into prominent sub-categories of spoofing apps: two that enable email spoofing, and two that spoof the device's MAC address.

Spoof – Burner Phone: 41 apps provide the ability to place anonymous calls or SMS messages, with many explicitly advertising for use in evading call blocking. These apps can be used both by abusers who intend to harass and by surveillance victims seeking to evade surveillance. These apps function as "burner phones" in that they provide phone numbers that can be used once and then discarded.

Spoof – Fake Call: 6 apps provide the ability to fake incoming calls or call logs, enabling users to spoof both the source phone number and caller-id. Fake incoming calls are often advertised as useful for getting out of "sticky situations", but other abusive purposes can be readily imagined.

Spoof – HTML: 5 apps enable the browser's rendered content to be altered, including changing the targets of HTML tags, which could be used to phish a victim.

Spoof – Image: 9 apps modify or create false images or videos, including face-swapping tools that can be used for impersonation attacks or for revenge porn [4]. Two apps enable images to be shared on WhatsApp for which the thumbnail provided by the app differs from the underlying image. Two others generate fake ID card images.

Spoof – SMS: 20 apps mask the true sender of SMS messages. Unlike burner-phone apps, the intent of many SMS spoofing apps is to pose as another individual. Many allow entire chains of text messages to be faked.

Spoof – Social Media: We found social media spoofing apps that impersonating senders and construct fake message chains. 7 of the 8 apps in this category spoof WhatsApp messages, while the 8th spoofs Facebook Messenger.

Spoof – Suppress Caller ID: 4 apps allow senders to fake or block caller ID information on the device where the app is installed. Most apps enable selective disabling or spoofing of caller ID on a per-call or per-sender basis.

Spoof – Thorough: 8 apps spoof in multiple ways. Most common were apps that combine burner-phone functionality with the ability to spoof caller-ID and voice spoofing. One app bundles fake email and SMS functionality.

Spoof – Voice: 10 apps use voice modification to mask identity or make a voice sound scary. Many of these are playful, but they do appear regularly alongside abusive apps.

Surveillance – Misc: Surveillance apps were the largest category in our data. While most surveillance apps fit cleanly into sub-categories, four apps were not part of any trend. These include two key-loggers, one app that is a viewer for keylogger installed on a PC, and one that logs touch input patterns.

Surveillance – Calls: These apps provide ongoing access to call histories or continual or selective on-demand recordings of phone calls without the victim's consent. 13 of the 18 apps in this category enable call-recording, with all but one claiming the ability to perform covert automated recording of calls. The remaining five provide ongoing access to call logs.

Surveillance – Camera: 15 apps turn on the camera and microphone, typically forwarding a stream to a remote device. Roughly half are marketed for covert use. Others re-purpose devices as security cameras or baby monitors, although PMI data suggests that many of these are also used for covert surveillance.

Surveillance – Location: 90 apps track location and little else, though location tracking is also offered by most thorough surveillance apps, making it the most common type of surveillance overall. Some of these apps are not covert and seem to be for child safety or business use cases, but most of the apps surfaced by CreepRank explicitly state, or strongly hint, that they are designed for covert tracking.

Surveillance – Microphone: 11 apps record the device's microphone, often to remotely turn on the microphone on a victim's device. Four apps use the microphone to enhance hearing, with titles like "Ear Agent: Super Hearing." While many of these apps market themselves for people with hearing disabilities, most encourage spying.

Surveillance – Screen: 5 apps allow the device's screen to be recorded, streamed, or snapshotted as their main purpose.

Surveillance – SMS: 9 apps focus exclusively on forwarding or snooping on SMS messages.

Surveillance – Social Media: Fully 105 apps enable continuous surveillance of social media accounts. Most prevalent are apps that enable access to multiple WhatsApp accounts on a single device, which can be used for benign purposes. However, malicious use of such apps is apparent in co-installation data and some of the apps themselves, as with "Clone Whatsweb Pro" which prompts, "Enter WhatsApp Victim's Device." Another group of apps provides users with digests of who viewed their social media profile.

Surveillance – Social Media Covert: 26 apps explicitly market their ability to surveil social media accounts covertly, such as by turning off indicators that abusers are logged into victim accounts and reading their WhatsApp messages.

Surveillance – Thorough: 90 apps provide multiple means of surveillance. App descriptions are often generic explanations of the app's capabilities without reference to illegal use cases, though in deference to app store policies or public pressure, some have since renamed themselves, as in the case of "GirlFriend Cell Tracker", which is now known as "Family Locator for Android." Suggested uses are most often anti-theft and parental supervision, but some mention remote control of a device or explicit "Spy", "Family", and "GirlFriend" tracking.

Tutorial – Hacking: 61 apps are hacking tutorials and provide device-hacking advice, tips, news, glossaries, and forums. "Spyboy" is both the most popular and most likely to be on devices with apps that appear to have abusive intent.

Tutorial – Rooting: 6 apps teach users how to root a device or actually do so. One such app, the "Kingo ROOT" app, is the 4th most prevalent app in the top 1,000, and is 5 times as prevalent as the other 5 rooting apps put together.

Tutorial – Settings: 16 apps provide guides and tools for changing Android "Secret Codes". These apps correlate strongly with hacking-focused tutorials.

Category	Sub-Category	Title	Package name
CallerID	Misc	Hello — Caller ID & Blocking	com.facebook.phone
	Location	Mobile Number Call Tracker	com.bhimaapps.mobilenumbertraker
Control	Use-Limitations	NetCut	com.arcai.netcut
	Hide-Icon	Hide App-Hide Application Icon	com.thinkyeah.apphider
Defense	Misc	Hidden Apps	soo.project.findhidden
	Anti-Harassment	Sms Retaliator	com.openwave.smsretaliator
	Anti-Surveillance	Oops! Applock	com.keybotivated.applock
		Incoming Call Lock - Protector	com.freesmartapps.incoming.call.lock.manager
Evasion	Alternative-Input	EVA Facial Mouse	com.crea_si.eviacam.service
	Hidden-Content	Smart Hide Calculator	com.ids.smartcalculator
	Steganography	PixelKnot: Hidden Messages	info.guardianproject.pixelknot
Harassment	Misc	Blue Whale Game	us.bluewhalegame.free
	Automation	AutoResponder for WhatsApp NEW	tkstudio.autoresponderforwa
	Bomber	Message Bomber -send 5000+ sms	com.logicup.messagebomber
		SMS Retaliator	com.openwave.smsretaliator
	Fake-Surveillance	Other Number Location Tracker	com.trackyapps.other_number_location_tracker
		SpyGo For Whatsa Prank	com.spygo.espiagowhatsa
		WhatsSpy VIP! PRANK	com.adm.whatsspyvipprank
Info-Extraction	Misc	Decrypto	info.valky.decryptor
	Dump	Inkwire Screen Share + Assist	com.koushikdutta.inkwire
	Fraud	Lodefast Check Cashing App	com.lodestar.checkcashing.lodestar
		Bank Card Validator	com.ndquangr.cardreader
		Credit Card Revealer	com.stb.cch
		Card Details Finder	carddata.carddatafinder.com.carddatafinder
	Hack-Tools	Droidbug Pentesting & Forensic FREE	com.droidbugfree.es
	People-Search	BaeList	com.baelist.www
	System	Mobile Sim and Location	apptrends.mobile_sim_and_location_info
None	Misc	Spy Mobile	it.linergy.spymobilewifi
	Android-Mods-VM	Never Uninstall Apps - SpaceUp	com.spaceup
	Communication	WhatsFriend for Whatsapp	com.bondrr.whatappfriends.chat
	Index	FileChef-OpenDirectory Finder	com.zqlabs.filechef
	Pay-Per-Install	Qbucks	com.company.qbucks
	Tutorial-Development	Master Android	net.androidsquad.androidmaster
	Tutorial-Misc	Mobile Software Flashing Vol—2	com.wMobileSoftwareCrackBoxall_4969181
Spoof	Misc	Fake Mailer: Send and Receive Fake Email	gq.fakemailer.fakemailer
	Burner-Phone	SMS Receive	com.smsreceive
	Fake-Call	Fake Call	caller.phone.id.fakecall
	HTML	Edit Website	web.dassem.websiteprank
	Image	Splitvid — Split Video Camera	com.niltava.javana.split
	SMS	Sending Fake SMS	br.com.ideatech.smsfakepro
		Spoof Text Message	com.spoofbox.spooftext
		Fake Text Message	com.neurondigital.FakeTextMessage
	Social-Media	Fake Chat Conversations	f.industries.fakemessages
	Suppress-CallerID	Caller id changer Sim	another.caller.id.changer
	Thorough	Fake Call	fakecall.fake.call.yo
	Voice	FunCall voice changer in call	com.rami_bar.fun_call
Surveillance	Misc	Hackers Keylogger	hack.hackit.pankaj.keyboardlisten
	Calls	Hidden Call Recorder	com.mrecorder.callrecorder
	Camera	IP Webcam	com.pas.webcam
	Location	Track a Phone by Number	com.androidaplicativos.phonetrackerbynumber
		Cheating Spouse Tracker	spouse_sms.tracker_app
		Find My Friends	info.com.dev.hkmobile.chatonline
		Where the hell are you?	com.where.the.hell.are.you
		Boyfriend Tracker Free	com.androidaplicativos.boyfriendtracker
	Microphone	Ear Agent: Super Hearing	com.microphone.earspy
	Screen	Screen Recorder No Root	eng.example.hatiboy.gpcapture
	SMS	SMS Forwarder	cz.psencik.smsforwarder
	Social-Media	Clone WhatsWeb Pro	clone.whatsapp.pro
	Social-Media-Covert	Unseen - No Last Seen	com.tda.unseen
	Thorough	GirlFriend Cell Tracker	com.omrup.cell.tracker
		Cell Tracker	es.cell.tracker.kids
		Family Locator for Android	com.omrup.cell.tracker
		Top Spy Apps	com.topgpapps.l
		Spy Mail	com.countmyapp.com.spymail
		Spy sms call controler	com.dspark.phone.modefind
		Control By SMS	smartmob.com.controller
		GirlFriend Cell Tracker	com.omrup.cell.tracker
Tutorial	Hacking	spyboy	info.androidhive.spyboy
		Cheating Spouse	com.eclipseboy.CheatingSpouse
		Cheating spouse tracker	catching.cheating.spouse
	Root	Kingo ROOT	com.kingoapp.apk
	Settings	Phone Secret Codes	com.neetu.ussdstrings

Table IX: For each category and sub-category of the codebook, we list all apps referenced in this work, or where no app pertaining to a category was referenced, we cite the app that was most prevalent in the 2017 data.