Discovery of Emergent Malicious Campaigns in Cellular Networks

Nathaniel Boggs
Columbia University
boggs@cs.columbia.edu

Wei Wang
AT&T Security Research Center
wei.wang.2@att.com

Suhas Mathur
AT&T Security Research Center
suhas@att.com

Baris Coskun
AT&T Security Research Center
baris@att.com

Carol Pincock
AT&T
carol.pincock@att.com

Abstract

The growth of Smartphones has bridged the telephony/SMS and the IP worlds, and this has resulted in new opportunities for financially motivated attackers. For example, some malicious campaigns in the cellular network aimed at extracting money fraudulently can do so even without any malware. Detecting and mitigating the variety of attacks in cellular network is difficult because they do not necessarily have a fixed 'signature', and new *types* of campaigns appear frequently. Further complicating matters, detecting a single malicious entity (a domain name, a phone number, or a short code) that is part of a malicious campaign, is usually not very effective, because the attacker simply moves to using another entity in its place. An effective strategy requires detecting all/most elements involved in the campaign at once. In this paper, we describe a system, based on ideas from anomaly detection and clustering, that aims to detect many different families of widespread malicious campaigns in cellular networks. The system reveals an entire campaign as a graph cluster which includes the various entities involved in the campaign and their relationship, such as malware download websites, C&C servers, spammers, etc. Using logs from both SMS and IP portions of the network for millions of users, we detect newly popular entities and cluster them to discover how they are related. By looking for cues of possible malicious behavior from any of the entities in a cluster, we attempt to ascertain whether a detected campaign might be malicious, providing valuable leads to a human analyst. Our system is live and generates daily clusters for human analysts. We provide detailed case studies of real, previously unseen families of malicious campaigns that this system has successfully brought to light.

ACSAC '13 Dec. 9-13, 2013, New Orleans, Louisiana USA

Keywords

Intrusion Detection, Mobile Phone Secuirty, SMS, Network Anomaly Detection

1. INTRODUCTION

Malicious activities in cellular networks form a complex landscape which can be quite different from traditional PC attacks. For instance, fraudsters can extract money from unsuspecting smartphone users by enrolling them in premium third party services without involving any kind of malware in the process [17]. As another example, botnet infected mobile phones can be used to distribute Short Message Service (SMS) spam messages which, unlike email spam, incur charges to victims' phone bills. Besides these unique attacks, mobile manifestations of more traditional attacks are constantly surfacing as well, such as malware-laced apps in app-stores [25] and drive-by download of mobile malware [6].

Network-based defense against this landscape of malicious activities requires non-traditional methods, mostly due to the unique relationship between smartphone users and service providers. More specifically, users pay for their mobile service, and hence, any security measure that may affect a user's service, such as blocking an SMS message or shutting down a suspected attacker's phone account, has to be first confirmed by a human analyst. Therefore, tools and algorithms that can help security analysts identify and confirm malicious activities in mobile networks are extremely crucial.

Motivated by this, in this paper we propose a network-based method to discover potentially malicious campaigns to be presented to a human analyst for confirmation. Since we aim to discover new families of malicious campaigns, we do not employ any 'signature' specific to a campaign. Our approach is based on anomaly detection on communication patterns, where we roughly consider an anomaly as a user communicating with an entity for the first time. Building a separate anomaly detector for each user is generally unfruitful because such a detector has too high a false positive rate, since humans frequently engage in communication with new entities. However, if many users have the same anomaly, this false positive rate can be lowered. This is useful because when a widespread malicious campaign occurs, one or more entities (a domain name, short code, or phone number) suddenly begins communicating with a significant

number of mobile phone numbers (i.e. victims) that they have not communicated with in the past.

Our method is a two-stage process. First, we identify entities (i.e. anomalous entities) which communicate with several phone numbers that they have never communicated with during a long, prior, training period. Despite this loose definition of an anomaly, often a large number of entities exhibiting the same anomaly indicates a potential malicious campaign. Therefore, in the second stage, we explore possible relationships between the anomalous entities by measuring the overlap between the sets of mobile users that they have communicated with. Such relationships among anomalous entities are essential to our method and can be captured by a *mutual contacts graph* (Figure 1), where nodes represent anomalous entities and an edge represents that a pair of entities share a significant fraction of users that have communicated with them. In a typical mutual contacts graph there are several connected components (i.e. clusters) each representing a potential malicious campaign. Finally we prioritize these clusters with respect to severity using various pieces of external information, such as publicly available blacklists, customer complaints, etc. and present the prioritized clusters to a human analyst for final confirmation.

We have implemented and deployed the proposed method in a large mobility network. We demonstrate its utility by a live study over several months. During this period we produced live daily reports of clusters suspected to be malicious, and identified a number of different attack campaigns (spam, premium number fraud, and suspected botnet behavior) some of which were subsequently shut down. In this paper we provide details of the most interesting malware campaigns that we unearthed. The contributions of our work in this paper are:

1) Signature free. Our approach is based on anomaly detection and hence allows for detection of several types of malicious activity without having to design a new signature for each new type of malicious campaign. Examples of attacks that can be detected include: domain flux botnets, premium number fraud [12], SMS-relays that defeat 2-factor authentication [4, 5] and multi-stage malware campaigns.

2) Scalable. Our approach is simple and able to scale up to a large mobile network with millions of subscribers.

3) Holistic picture. Rather than just a single malicious entity in isolation, our method reveals entire campaigns as a cluster with most of its malicious entities such as mobile phone SMS spammers, malicious domain names, and short codes. Blocking or shutting down a single phone number or domain name does almost no harm to the attacker, whereas moving swiftly against multiple pieces of the attacker's infrastructure is more likely to end the campaign.

4) Real malicious campaigns. We present the details of real cases of malicious campaigns that we discover using our method, providing insights in the modus operandi of cyber criminals.

2. RELATED WORK

In general, network-based anomaly detection systems try to characterize normal network behavior and produce an alarm when the behavior deviates from normal. Although in principle, anomaly detection systems are able to detect previously unseen attacks, unfortunately characterizing normal network behavior is notoriously difficult. Therefore, network-based anomaly detection systems often suffer from

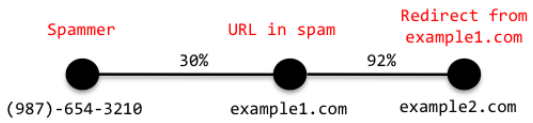

Figure 1: In the mutual contacts graph, nodes represent entities, and an edge between two entities represents a significant overlap between the sets of users that communicate with them. In the example above, an SMS spammer sends numerous messages to many subscribers. A significant fraction of spammed users click on the URL within spam messages, hence the edge between spammer and URL. Finally almost all users who click on the spam URL are redirected to a new domain, hence the edge between spam URL and new domain name.

high false positive rates. There have been several attempts to achieve anomaly detection systems with lower false positive rates such as by manually characterizing normal behavior using protocol specifications [19] and by incorporating application level knowledge [13]. Nevertheless, anomaly detection systems have evidently never been able to achieve acceptable error rates. Despite the plethora of anomaly detection methods proposed in the literature [21], one can hardly find a deployed anomaly detection system in practice [20]. To achieve a practical anomaly detection system, we employ a simple key strategy different from existing schemes. That is, an anomalous behavior is taken into account only when large number of network users exhibit the same anomaly thereby reducing the false positives.

Aside from anomaly detection, there are several other areas which are related to our work. As mobile phones and PCs have much in common, many of the defensive techniques have direct application to mobility networks. Domain reputation blacklists based on user reports, spam detection, and malicious activity provide an inexpensive but time-lagged detection method to prevent continued abuse. By limiting an attacker's use of particular domains, these blacklists increase the cost to attackers, forcing them to change domains, but rarely do more than slow the attacks.

Network based malware detection often focuses on botnet detection. While there are numerous botnet detection schemes proposed in the literature, clustering analysis or similarity-based methods are most closely related to our work. In [11] and [24], authors make the observation that members of a botnet exhibit similar network level characteristics due to underlying common malicious behavior, therefore clustering analysis of several features extracted from observed network traffic would group the member of a botnet together. In [8], the authors propose to detect P2P bots by applying a graph-clustering method on a mutual-contacts like graph extracted from network flow records. In [14], the authors analyze DNS data from cellular networks to characterize the amount of malicious activity in cellular networks. They find a malicious campaign that ceases to operate long before the malware associated with it is discovered, and recommend that "network-based countermeasures may be useful in the identification and mitigation of future threats".

While attacks in mobility space are getting more sophisticated [25], current defenses in the mobility space for the most part mirror early PC defenses. For instance, in order to detect SMS spam, volumetrics and content hash/signature

Term	Definition
Entity	Domain name, phone number or short code.
User	A 10-digit phone number with data plan.
High-degree node	An entity that communicates with a significant number of users.
Normal node	A high degree node with many users communicating with it in the training phase.
Anomalous node	A high degree node with many new users communicating with it in the testing phase, but not in the training phase.
Communication graph	A graph in which vertices represent entities and edges represent communication.
Mutual contacts graph	A graph in which a vertex represents a high-degree node and an edge between two vertices represents a significant fraction of shared users.

Table 1: Terminology used in this paper

matches [15] are used at the network level. In [10] authors propose a clustering based SMS spam detection method, where they observe that SMS spam messages have similar contents therefore they are likely to be clustered in a random subspace. In [22] authors propose exploiting message senders' temporal characteristic (how regular they send message) and social characteristics (who sends to whom) to identify spammers. However, computing such characteristics often runs into scalability problem when large networks are concerned. To mitigate such scalabilty issues, in [9], authors propose an efficient SMS spam campaign detection technique which quickly identifies unusually high number of similar contents transmitted within the SMS network. While effective, these techniques focus on detecting spam messages and spammers and cannot reveal the further stages of the overall malicious activity. Finally, there is important prior work [**?**] on analyzing communication patterns using graphs to detect fraud in cellular networks. [**?**] introduces guilt by association (which we build upon), but utilizes the actual communication graph for its analysis (whereas we work with the mutual contacts graph), and aims to detect simpler fraud, such as phone accounts created with the intention of not paying their bills, rather than malicious campaigns.

3. SYSTEM DESIGN

3.1 Overview of Attacks and Defenses

A typical widespread attack in a cellular network has at least two communication components (Figure 2):

1) The initial infection or luring. The initial attack may occur through the medium of a spam SMS message containing a URL, an app store, or perhaps a hacked website. These websites may serve anything from drive-by download malware attempting to infect phones to social engineering attempts to trick a user into voluntarily downloading malware or signing up for a premium rate SMS service under false pretenses.

2) The subsequent behavior of those users that get infected or fall for the attack. This might include communicating with a command and control server / drop-server, or an SMS shortcode, relaying all SMS messages to a certain phone number, or perhaps acting as an SMS spammer itself.

For example, in the GGTracker malware campaign [18], Android users are tricked into downloading an app from a website resembling the Android app market. The malware then registers the victim for premium subscription services without any action on the user's part. Signing up for such a service normally requires answering a question or device's own phone number and entering a PIN code received via SMS. However, the malware performs this transaction automatically, without the user's knowledge. The premium service bills the carrier which reflects the charge in the user's monthly bill.

There are two key observations on these communication components of malicious campaigns:

- Almost always, there is more than one entity involved in a campaign, e.g. a 10-digit number that sends a URL in an SMS spam, and the domain name in the URL that users click on.
- The entities involved often change over time. This is because some of the entities may have complaints filed against them, and are taken down. But the attacker simply begins using a new entity of the same type. For example, if a mobile phone number is reported to be sending spam and is shut down, then the attacker can simply start sending spam from another number.

Based on the above two insights, we focus on a specific type of anomalous pattern: the existence of multiple entities that have recently become very popular (i.e. they have recently begun communicating with a large number of users), *and are also* related to one another via a non-trivial overlap between the sets of users they communicate with.

Figure 3 shows a high level overview of the steps involved in detecting suspicious campaigns. Each step in this chain winnows down the data, keeping anomalous traffic and dropping irrelevant traffic, until it is small enough for a human analyst to investigate. First, we gather the data and match users in both data sets (SMS and IP) in order to have a complete picture of the traffic for these users, since many attack campaigns involve traffic in both the SMS and IP worlds. Then, using data from an initial time window, we discover domains and numbers that are already popular (training). In the testing phase, which corresponds to a subsequent time window, these already popular entities are excluded, and we calculate how many and which distinct users each entity communicates with. We then take the most popular of these entities and calculate how many users each pair has in common. In this way we infer patterns that indicate similar sets of users connecting to multiple newly popular entities. Using the overlap between the sets of users that communicate with each newly popular entity, we cluster the entities together using a graph clustering algorithm.

A cluster is the final product of our algorithm and the canonical unit of suspicion in our system. That is, given our traffic-centric viewpoint, a malicious campaign *is* the

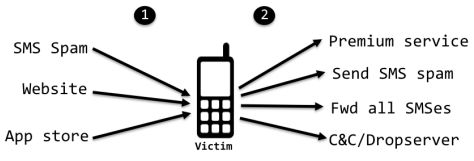

Figure 2: An attack campaign life cycle. (1): Device gets infected through some channel, (2): Infected device communicates with other entities as part of the attack.

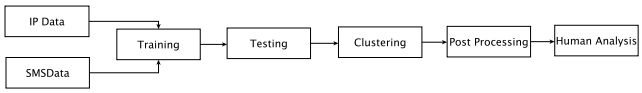

Figure 3: The overall system as a sequence of operations starting with raw data on the left.

set of entities used to perpetrate it. By the time we form a cluster, we have reduced the quantity of data down by a factor of approximately $\sim 10^4$ nodes and by a factor of $\sim 10^5$ edges. In this form, the data is therefore suitable for consumption by a human analyst (10-100 clusters), who then decides to further invstigate or discard each cluster. Note that, a cluster may represent a malicious campaign, or a benign infrastructure. A multi-stage attack is revealed as one cluster because the same set of users will go through different stages such as users get spammed are also the ones that communicate with the malicious domains and premium numbers.

As an aid to the analyst, we enhance each cluster by adding relevant information. For each entity in a cluster, we include the number of users that communicate with that entity during the test window. We also include the amount of correlation between two entities in a cluster on the graph. Finally, we prioritize clusters using external information, such as whether some entities of a cluster appear on a public blacklists or some numbers are reported as spammers by many mobile users through the 7726 short code[1]. Our system is running live to generate daily clusters for the most recent one week period as the test window.

3.2 Data Sets

Our raw data consists of SMS messages, and IP data belonging to a large US cellular carrier. The system only utilizes *who-talks-to-whom* information for both SMS and IP data, no content data is used. All phone numbers are anonymized to protect privacy. Any phone numbers shown in this paper are anonymized to 64 bit tokens. Any results in this paper involving traffic volume are scaled randomly to prevent revealing true traffic volumes of any kind, but still maintain patterns involving relative changes in volume.

Each record in SMS data is simply a time-stamped tuple indicating the source and destination phone number. A phone number can be either a 10-digit numbers or a shortcode (used for SMS based services such as checking TV voting, movie times, etc.). In each record, either the source or the destination (or both) must be subscribers of the carrier. Due to volume constraints, only a geographically based sample is used. IP data, using the same geographically based sample, consists of timestamped tuples indicating the 10-digit phone number and the 2nd level domain name visited. In order to focus on data-enabled phones, we use SMS data only for users that have at least one record in IP data. In an average one week period, our sample provides about ~ 150 Million communication edges, involving ~ 40 Million unique entities, of which ~ 10 Million are 10-digit phone numbers.

3.3 Training Phase

In order to pick out anomalous traffic patterns, our dataset is divided into two parts by time: a training phase, 1 month in duration, and a test phase, which is a sliding window of duration one week, that appears chronologically after the training phase. The purpose of the training phase is to build a notion of what constitutes normal traffic patterns.

3.3.1 Why not edge-based training?

Here, we model the appearance of edges between nodes in the communication graph. If an edge is present in the training set, then it is considered normal, and is therefore dropped if present in the test set. While this approach seems undoubtedly important for modeling normal user interactions, unfortunately, it is incapable of identifying high-degree entities that are already popular in the training phase. For example, since entities like `cnn.com` and `yahoo.com` already have high degree in the training set, we would not want to retain them in the test set. Unfortunately, there are a large number of users that have edges to `cnn.com` in the test set but not in the training set. That is, there are always new users communicating with such entities. We empirically found this pattern to hold for a large number of entities that already have high degree in the training set. Therefore the edge-based training strategy would end up selecting entities that should not be selected, thereby adding a lot of noise to our mutual contacts graph. As an alternative, we consider node-based training below.

3.3.2 Node based training

Since we are interested in high-degree nodes, in practice we employ a training method that focuses directly on identifying high-degree nodes, rather than edges. Entities that are high-degree in the training phase can be considered to be already popular, and can therefore be discarded from the test dataset along with all their edges. The task of testing then simplifies to one of finding entities with high degree in the remaining communication graph. In practice there is another important issue to consider: shortcodes and phone numbers can change ownership over a period of time, while domain names typically do not. This implies that we should be less aggressive in whitelisting phone numbers and short codes. Further, as domain names are rarely reused and malicious domains once blacklisted become less useful to attackers, we are able to train purely on whether a domain is seen previously in this initial prototype. With this in mind, in the end we perform the following training steps:

1) Domain names: All domains that appear in the training window are dropped from test set.

2) 10-digit numbers and shortcodes: Only a select known benign services (gateways, etc.) is whitelisted. All other numbers are kept.

While the latter does not remove benign high degree entities, we do not expect high degree benign entities to form clusters of significant size with other entities. Therefore, keeping almost all numbers allows us to retain both types of high degree nodes, benign and malicious, at the expense of deferring the detection to the clustering stage. In the future, as attackers adapt to poison these training methods, we will explore additional more sophisticated training approaches.

3.4 Testing Phase

During the testing phase, the entities identified in the training phase as normal are dropped from the test dataset, and the remaining communication graph is used to compute the mutual contacts graph. As introduced in Section 1, the mutual contacts graph is a graph in which nodes represent

[1]GSM Association www.gsma.com/technicalprojects/gsma-spam-reporting-services

abnormal high-degree entities in the testing phase, and two nodes are connected by an edge if there is significant overlap between the sets of users that communicate with the two entities. The intuition behind building the mutual contacts graph is that the users that end up connecting to a later part of an attack campaign such as a fraudulent premium number or botnet command and control will also have connected to the same prior part of the attack campaign like receiving a spam SMS message. In this way we can link multiple stages of the same campaign in a single cluster, such as the number sending spam and premium numbers involved in the conversion step.

To build the mutual contacts graph, the relationships between these anomalous high degree nodes are analyzed. Specifically, we compute the Dice association coefficient [23] between every pair of high-degree nodes. The Dice association coefficient $D(a, b)$ between two nodes a and b is defined as:

$$D(a, b) = \frac{|A \cap B|}{\min\{|A|, |B|\}}$$

where A and B represent the set of users that communicate with nodes a and b respectively, and $|\cdot|$ represents the cardinality of a set.

Complexity and Scalability To identify mutual contacts between anomalous nodes, we need to maintain a full set of entities that each node communicates with. This poses scaling issues. The size of memory needed is at worst quadratic in the number of nodes (N), but in practice is linear as nodes average few unique contacts. Computing the number of mutual contacts between nodes requires a set intersection computation between every pair of nodes, with runtime of $O(N^2)$. To mitigate with this issue, we only consider the K highest degree nodes in the mutual contacts graph. To compute the number of mutual contacts between these K nodes, for each one of N entities, we determine the subset of the K entities that it communicates with and increment the number of mutual contacts between K_i and K_j if both K_i and K_j are in the subset. This reduces the complexity to $O(K^2 * N)$, which is $O(N)$ for $N >> K$.

In our daily reporting experiments, we use a test window of duration one week, and we slide this window over time, one day at a time. We choose K to be in the order of $50,000$, with the lowest degree node selected having a degree of ~ 50 on average. Processing around 10 million unique users and their connections takes less than 40 GB of RAM in our prototype system and we can generate a daily report overnight. The system can be easily extended to include more high degree nodes for correlation if the hardware and computing time permit.

3.5 Clustering

The clustering phase involves partitioning the mutual contacts graph into clusters, such that each cluster represents a suspicious group of entities involved in the same malicious campaign. Edges in the mutual contacts graph are weighted by the Dice association coefficient as described in Section 3.4. We partition this graph in two stages. First we break any edges that have a Dice coefficient of less than 0.1, or have an absolute number of shared users of less than 20. These thresholds represent the 99^{th} percentile of all edges in the mutual contacts graph.

Then, we use the work of Blondel et al [7] to perform graph clustering on the remaining graph. Blondel optimizes a quantity called *modularity*, where the modularity of a graph partition is a scalar value between -1 and 1 that measures the density of links inside clusters as compared to links between clusters. Thus it finds high modularity partitions of graphs in order to split clusters that would otherwise only have a weak link between two highly connected groups of nodes. The nodes remaining connected to each other after this process form each of our clusters. Most clusters are of size 2 or 3 (distribution of cluster sizes discussed in Section 3.6.3). We currently focus on clusters of size > 3, mostly because small clusters of size 2 and 3 remain active for a very short period of time (Section 3.6.3). Human analysts can prioritize analysing big clusters and investigate small clusters when time permits or additional activity proves them suspicious.

3.6 Post processing

Once we have arrived at suspicious clusters, we are faced with the final task of classifying each cluster as malicious or benign. The final decision is left to a human analyst, but to make the task as automated as possible, we add additional contextual information to better inform the analyst.

3.6.1 Temporal traffic pattern.

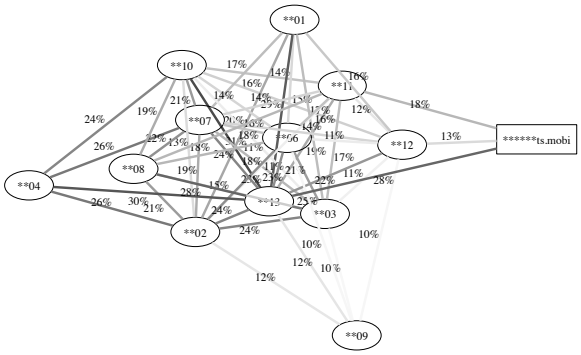

Figure 4: A benign cluster consisting mainly of SMS shortcodes for voting on a TV show. Figure 5 shows the regular traffic patterns produced by the entities in this cluster.

The temporal traffic pattern of nodes in a cluster can provide information on whether it is likely to correspond to a programmed event or show (and therefore a benign cluster). As an example, consider the cluster we discovered shown in Figure 4. We found in Figure 5 that the traffic belonging to the entities in this cluster has peaks occurring at a specific time once a week. It turns out that this cluster corresponds to a popular TV show that involves SMS voting and the peaks occur exactly during the show hours. This pattern by itself cannot be used to label a cluster as benign for sure – since a similar pattern might be utilized by a malicious campaign, such as data exfiltration only at night time or periodic botnet command and control – but it can help an analyst direct her investigation.

3.6.2 Label Known Malicious Nodes

As attackers use domains and premium shortcodes for many attack attempts, and these components of widespread attacks are almost always eventually detected, labelling our clusters using data from existing blacklists can give analysts

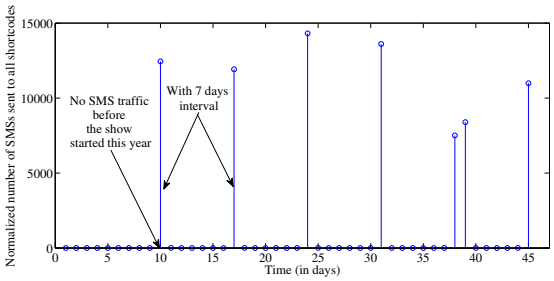

Figure 5: Number of SMS messages sent to each of the short codes in Figure 4 over time. There are periodic traffic volume peaks during the show time each week. Absolute values of traffic volumes on the Y-axis are obscured by scaling uniformly by a random number.

important clues about the nature of a cluster. Blacklists complement our approach by providing some partial ground truth. Some weaknesses of blacklists such as lagging behind other detection methods and providing no relationship between malicious nodes, are complemented by our clusters, which can group a large part of an attack campaign while quantifying the relationship between abnormal nodes. We use a variety of blacklists, both of SMS numbers based on user reports [3] as well as third party domain name blacklists.

3.6.3 *Cluster Size and Change Over Time*

The size of a cluster and whether it changes in composition over time can provide clues into whether it is malicious. This is because malicious campaigns frequently need to put new numbers and domain names into use as old ones get taken down or blacklisted. On the other hand, legitimate services are more likely to keep the same domains and numbers as users become familiar with them. In this section, we compare the clusters from a given window to clusters from a previous test window in order to highlight clusters that evolve over time.

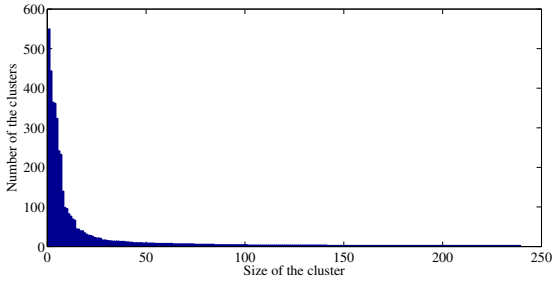

Figure 6: Cluster size distribution.

Figure 6 shows a histogram of the average size of clusters over test windows. In total, approximately 10 clusters are bigger than one hundred high degree nodes while 30 clusters have the size between ten and one hundred, and around 50 clusters have a size of between five and ten nodes. Most clusters have less than five nodes.

A *new cluster* is a cluster which has no overlap with any clusters in the previous sliding window. Similarly, an *obsolete cluster* is one which has no overlap with any clusters in the next test window. Any clusters which have non-zero overlap with any cluster in the previous window are *active clusters*, which means the nodes in those clusters are still high degree nodes with users regularly communicating with them. For active clusters, we measure the similarity between two versions of the same cluster in different test windows using the Jaccard similarity coefficient.

Figure 7 shows the Jaccard similarity coefficient ($= \frac{|A \cap B|}{|A| \cup |B|}$) for clusters in two adjacent sliding windows. If one cluster overlaps with two clusters, which means that the cluster is regrouped into two, the larger similarity coefficient is chosen. The figure shows that on an average day 85% of the clusters do not change from one test window to the next. Almost all of the 15% of clusters that change daily are of the size greater than 5. Since only around 100 clusters are of the size greater than 5, roughly ∼15 clusters change each day.

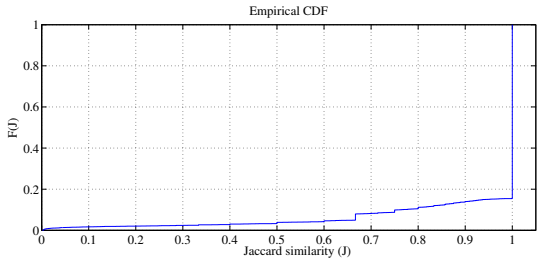

Figure 7: Average Jaccard similarity of the same clusters between two successive test windows (85% of clusters remain identical).

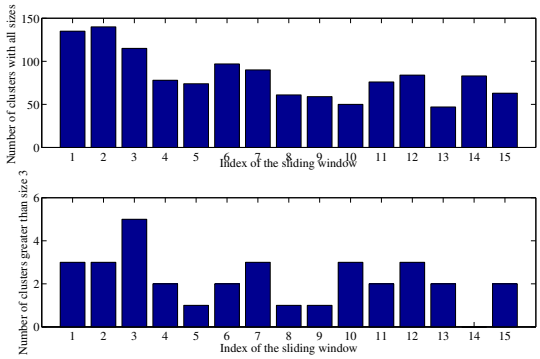

Figure 8: The number of new clusters (top) and the number of new clusters of size > 3 in 15 successive sliding test windows.

Figure 8 shows the number of new clusters generated for several successive sliding test windows, along with the number of clusters of size > 3. Additionally, 96% to 97% of obsolete and new clusters are size of less than or equal to 3 nodes. Only ∼3% of new clusters are of size > 3 nodes. Of the average ~ 100 new clusters that are generated daily, only ~ 3 are large clusters per day.

Therefore, with approximately ~ 3 new and ∼15 changed clusters per day, the system generates a reasonable amount of work that a human analyst should further investigate.

3.7 The Role of the Analyst

All the steps up to this stage have involved automated processing to winnow data down as much as possible, while

retaining suspicious activity. The role of the human analyst is to decide whether a cluster presented to her is malicious or benign. Figure 10 shows an example of a cluster presented visually to an analyst. We have found that our post-processing steps add significant value to the decision making process of an analyst.

At times, it is relatively easier to determine that a cluster is not malicious by noticing patterns in the domains names involved in the cluster. For example, some benign cluster that we have clearly identified as such include: "current event clusters", "political clusters", and "holiday clusters". After the analyst has removed what appear to be benign clusters, a finer level of manual analysis is required for suspicious clusters. The analyst can scrutinize the domains contained within the cluster and identify malicious or fraudulent intent. As with most malicious activity investigations, a certain amount of fine grained research into the event or activity is required.

4. EVALUATION & CASE STUDIES

Traditional evaluation of a detection algorithm calls for a true positive and false positive rate with a calculation of the tradeoff between them, typically visually illustrated in a Receiver operating characteristic (ROC) curve. However, the difficulty in establishing ground truth when dealing with real data and our system's role as a tool for analysts rather than a standalone blocking/detection mechanism forces us to look for other evaluation mechanisms. At the network provider level, the process for blocking abusive users such as those involved in spam and malware campaigns centers around analysts who gather sufficient information before action is taken to block.

With this reality in mind, we want to answer two questions: Does our system detect widespread attacks in the clusters it produces, and do we miss some widespread attacks? To answer the first question, we have confirmation that two premium numbers in our large example cluster were blocked due to fraud (details in section 4.1), and we also successfully detected the resurge of the Android "NotComapatible" malware on the first day that the malware broke out (details in section 4.2). Furthermore, by comparing our clusters to user reports such as SMS spam and domain reputation, we observe that the number of entities in clusters being blacklisted increases over time. This demonstrates that our system detects malicious entities ahead of the existing blacklisting systems.

For the second question, we note that there have been a few malware campaigns that were published in media reports. Two campaigns have been successfully detected. But the system did not detect a small campaign where malware infects devices and controls them via commands to send SMS spam [16]. The reason is that the malware download sites are high degree nodes without enough overlap to be clustered, and the C&C domains do not appear due to a low volume of infected users. If the campaign had increased in size, our algorithm would have detected the correlation in the initial infection vectors. No other widespread mobile malware campaigns are reported in the news and other investigated methods during our experiment period.

Growth in blacklisted members. One way to check whether our system works is to check the members of each cluster against several public and private blacklists, and see if the fraction of entities in a cluster that are blacklisted

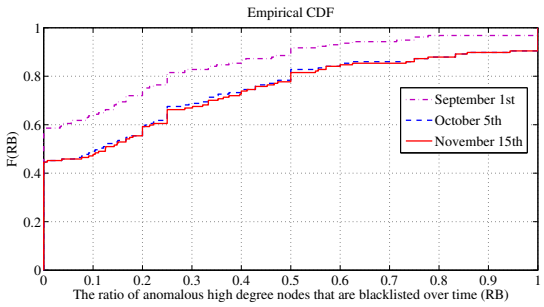

Figure 9: The number of anomalous high degree nodes that are blacklisted increases over time.

somewhere goes up with time. We use 3 sources of blacklists:

- For SMS numbers, we check against 7726 data, which is a feed of user reported SMS spam and an internal list of numbers sending spam URLs. Both of these data sets are noisy so we only match against numbers that are highly likely to be spammers.

- For domain names, we check our internal URL spam list and Web of Trust's score [2]. Again, as user reports are noisy we only consider high confidence listings, in this case a score of 10 or less on Web of Trust, which means the domain is either blacklisted by Spamhaus [1] or has many user reports.

Figure 9 shows the ratio of cluster members that are labelled as malicious by the blacklists above. It can be seen that the number of blacklisted members increases over time. Also some of our clusters are identified as 100% malicious by the blacklists, which provides further confidence that we are detecting malicious campaigns.

Furthermore, we ran queries against domain name and IP address lists known to belong to botnet C&C servers, to check if any known widespread botnets, are active in our cellular network's IP data, but were missed by our detection method. After checking against thousands of active fastflux domain names, we only came up with a dozen users sparsely communicating with 3 blacklisted domains. As a result, we concluded that we did not miss any significant known botnet activity in the cellular network. Finally, from public media reports and internal investigation, we found only evidence of the one small botnet described earlier and no widespread mobile botnet incidents in the cellular world, so to the best of our knowledge, our system did not miss any well discovered widespread attacks during the reported period.

In the remainder of this paper, we describe a few case studies centered around the most interesting malicious clusters encountered by us to date in the cellular network. The purpose of these case studies is to give the reader a taste of the types of malicious campaigns we unearthed in the cellular network. In each case, we describe why the cluster piqued an analyst's interest, and the manual analysis conducted to dig deeper into ascertaining the true nature of the cluster.

4.1 SMS Giftcard Scam

Suspicious cluster During July - Sept 2012, we detected a cluster consisting of 10-digit phone numbers, SMS shortcodes and domain names. This cluster continued to evolve

constantly throughout this period. Over time, several entities in this cluster appeared in domain name blacklists and the 7726 blacklist. Figure 10 shows this cluster evolving over time starting with the day it was first detected.

The very fact that this cluster contains at various times, phone numbers, short codes and domain names, is interesting, because it suggests activity that involves all three types of entities. Several interesting changes occur in the cluster from August to September. The cluster first grows in size, by adding several 10-digit phone numbers, which are soon blacklisted in the 7726 blacklist as SMS spammers. New domain names continually appear, while old ones disappear from the cluster. One specific domain name seems to be at the center of a star shaped region of the cluster, suggesting a series of redirect that lead up to it. A shortcode then appears and remains in the cluster, strongly correlated with the cluster center. Successive incarnations of this cluster are visualized in Figure 10. Figure 11 shows the Jaccard similarity between successive versions of the cluster across a 1 month time frame.

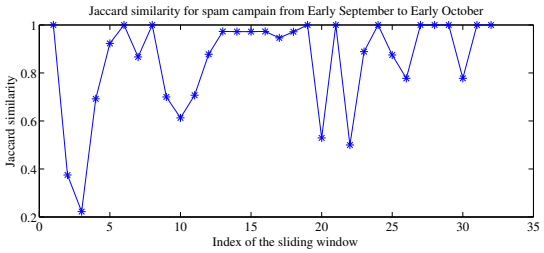

Figure 11: Jaccard similarity across successive days for the giftcard scam cluster from early September to early October.

Manual analysis In order to confirm the malicious nature of this cluster, our analyst performed a detailed analysis of the attack starting with one of the URLs present in the SMS messages. We found the following malicious campaign.

A carefully crafted URL with a well-known brand name in the 3rd or 4th level domain name position is placed into an SMS spam message promising a free gift card from that brand. This initial URL redirects the user to another URL, where the user is asked to enter the "winning code" provided in the SMS spam message along with their phone number and address in order to "claim their prize". Next, the user is redirected to yet another URL where they are told they must complete a "survey" to receive the gift card. This "survey" actually elicits the user to participate in receiving one of the products advertised on the site. Items like a book of the month club or magazine subscriptions are on offer, however the user must pay to participate in these services in order to receive their free gift card. Further investigation suggests that a significant fraction of users who visit these pages are automatically signed up for a premium rate service that utilizes a short code appearing in our cluster. We suspect that the step of entering a phone number and PIN code triggers this sign up.

We can see from the clusters that spammers and the short domains names are highly correlated, as are the short domain names and redirected domains. We also see that the number of entities labelled as malicious over time increases. We also observe in Figure 10 that the campaign loses its power in early October. So the node which represents the premium number in previous figures disappeared from the cluster because it got blocked by the service provider and was not a high degree node any more.

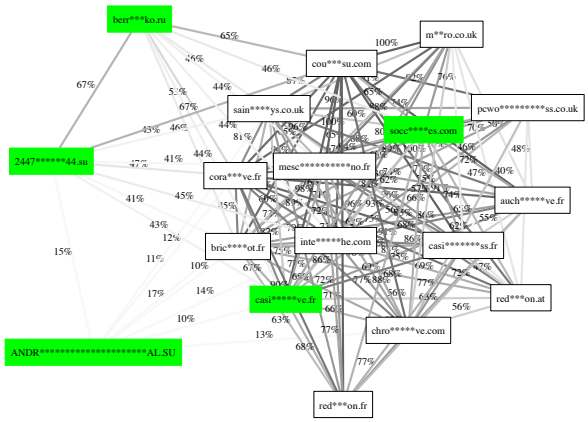

Figure 12: "NotCompatible" Android Trojan malware cluster.

Android**********ecurityupdates.su	244*********244.su	Berr*****ko.ru
Androidcloud**********update.su	244*********133.su	Gala*****ki.ru
Android**********portal.su	243*********334.su	Marr*****ko.ru

Figure 13: Three groups of suspicious newly created domains, referred to as the Android security updates set, the number set, and the "ru" set, are highly correlated in one cluster.

4.2 NotCompatible Android Trojan

Suspicious cluster One of the clusters that the system recently generated consists of a number of newly registered domain names (colored green in Figure 12). Although none of the domain names were found to be blacklisted anywhere, a couple of observations make this cluster suspicious to a human analyst. First, some of the names of some newly created domains (examples of suspicious domain names are shown in Figure 13) seem suspicious because they are related to Android security updates, but they are not related to Google based on domain registration information. Another group of domain names in this cluster consists purely numeric digits. Finally, these two sets of domain names are correlated in the mutual contacts graph, and the numeric domains are also highly correlated with another group of newly created domains, which end in `.ru` as the top level domain (Fig. 12).

Manual analysis Based on the observations above, this cluster was suspicious enough to warrant manual analysis by a human analyst. We discovered the following malicious campaign.

1) Users receive an email spam, primarily spread from hacked email accounts.

2) When a user clicks a URL in the spam from a browser on a non-Android operating system (Windows, iOS, etc.), the user is redirected to a fake Fox News weight loss article. But an Android browser is redirected to a "Android security

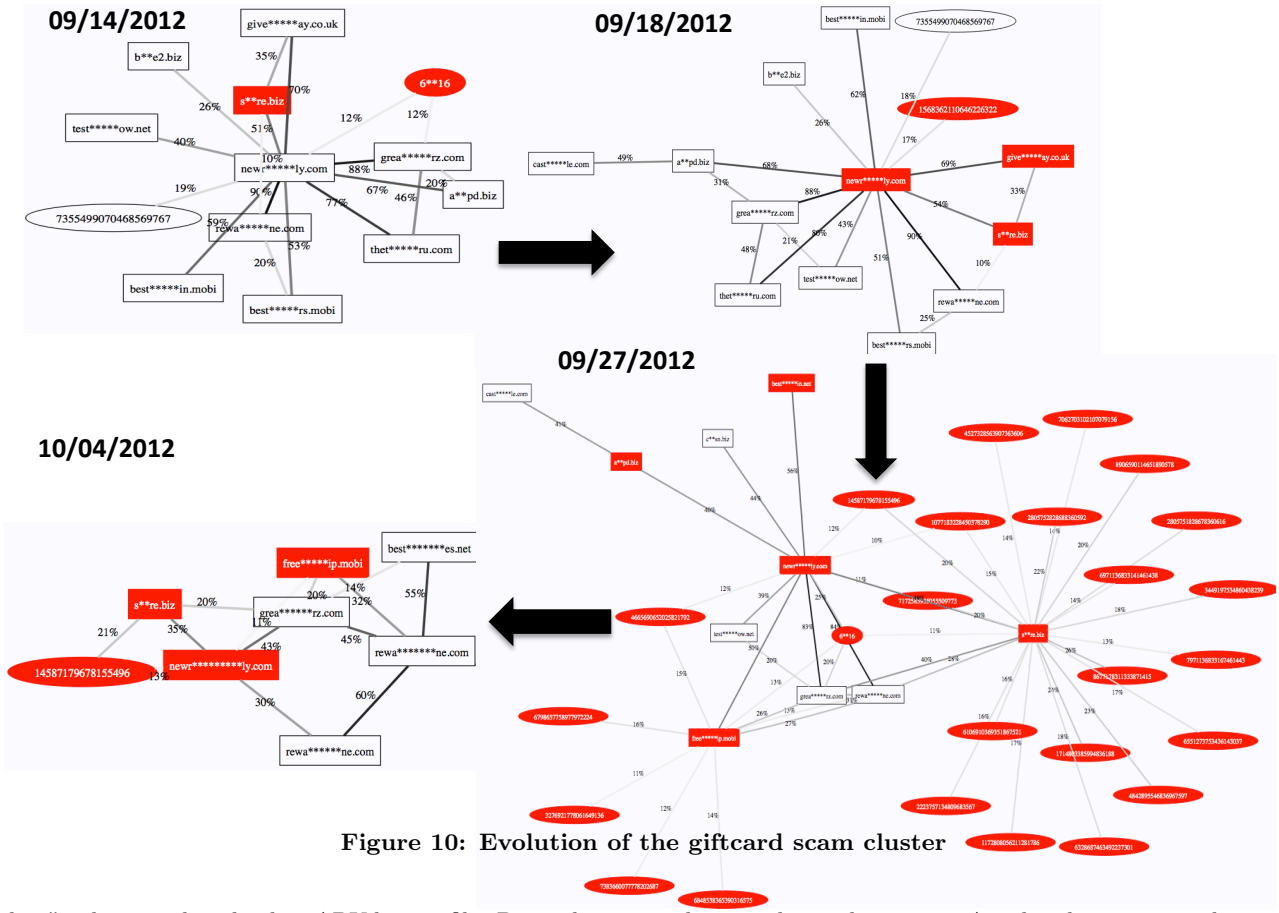

Figure 10: Evolution of the giftcard scam cluster

update" website to download an APK binary file. Depending on the user's Android OS Version and browser, the user is or is not prompted about the APK download.

3) The APK file is a trojan and runs as a background service. Once installed, the trojan acts as a proxy, thereby allowing its owner to transmit and receive network data through the infected device. The infected device persistently connects to a C&C server to maintain the communication.

4) The numeric domains and `.ru` domains resolve to the same set of IP addresses, which are where the C&C servers are hosted.

We confirmed that the cluster successfully renders a holistic picture of the entire malicious botnet campaign. The domain names in the URLs in spam emails, the websites where the Android APK binaries were hosted (Android security update domains), and the final C&C servers that the malware made connections to (both the numeric domains and the `.ru` domains), are all present in a single cluster.

5. LIMITATIONS AND FUTURE WORK

Limitations. There are a number of limitations of our system. It does not deal with targeted or very small scale attacks, which are significantly harder to catch because they do not create correlation patterns of the type we exploit to build clusters. Our clusters also do not attempt to catch attack campaigns that are based on a single malicious entity (we believe this is somewhat unlikely in real campaigns). Since we focus at the network layer, we would miss malicious entities situated in the application layer such as using a Twitter account for C&C although we could still detect the initial attack vector. Another limitation, inherent to anomaly detection, is that if attack traffic increases in volume very slowly, it will not be caught and will eventually be used as historical training data for a later testing time window. Therefore very slow attacks can evade detection, but would cost attackers extra resources as the attack is slower and blacklists although lagging will eventually block it. Finally, we do not address the legitimate app stores as a source of the initial attack vector – the domain name associated with it would be whitelisted as an entity which receives a large amount of traffic during the training phase – we believe keeping major app stores clean of malicious apps is a problem that the owners of the app stores have a strong incentive to enforce.

Future work. In future work, we plan to add additional sources of side-information in characterizing clusters, in addition to looking for blacklisted entities. For example: the age of domain names, phone numbers and short codes, information extracted from `whois` queries on the domain names in the cluster, etc. We also hope to perform a longitudinal study and evaluate our performance against new mobile threats as they occur. To further evaluate the effectiveness of our approach, it would be useful to have a careful long term study of the impact on analysts of using our tool, compared to using only existing tools. We are also interested in applying our approach to the detection of PC botnets, to help automate the correlation between fast flux domains.

6. CONCLUSION

We have built a system that allows us to use cellular network traffic to identify potential malicious campaigns. Our system combined with additional outside data gives a holistic view of the interconnected pieces of an attack campaign. Our approach is a complementary layer of defense useful in mitigating the effectiveness of widespread attack campaigns such as spam, premium number fraud, and malware. By identifying all components of an attack campaign rather than just the initial numbers sending spam, we can greatly increase the cost to the attacker by not only shutting down the spamming phone numbers, but also shutting down more expensive resources such as premium shortcodes. By tracking attack nodes over time, we can effectively identify any resurrection of the same attack campaign unless an attacker completely restarts with brand new numbers, domains, and shortcodes, which would greatly raise the cost for attackers.

7. REFERENCES

[1] The spamhaus project. `http://www.spamhaus.org/`.

[2] Web of trust, safe browsing tool. `http://www.mywot.com/`.

[3] Wikipedia article on mobile phone spam – countermeasures. `http://en.wikipedia.org/wiki/Mobile_phone_spam#Countermeasures`.

[4] Fortinet security blog: Zeus in the mobile (zitmo): Online banking5s two factor authentication defeated. `http://blog.fortinet.com/zeus-in-the-mobile-zitmo-online-bankings-two-factor-authentication-defeated/`, Sept 2010.

[5] S21sec security blog: Zeus mitmo: Man-in-the-mobile. `http://securityblog.s21sec.com/2010/09/zeus-mitmo-man-in-mobile-i.html`, Sept 2010.

[6] Lookout security alert: Hacked websites serve suspicious android apps. `https://blog.lookout.com/blog/2012/05/02/`, May 2012.

[7] Blondel, V., Guillaume, J., Lambiotte, R., and Lefebvre, E. Fast unfolding of communities in large networks. In *Journal of Statistical Mechanics: Theory and Experiment* (2008).

[8] Coskun, B., Dietrich, S., and Memon, N. Friends of an enemy: Identifying local members of peer-to-peer botnets using mutual contacts. In *In Proc. of the 26 Annual Computer Security Applications Conference (ACSAC)* (2010).

[9] Coskun, B., and Giura, P. Mitigating SMS spam by online detection of repetitive near-duplicate messages. In *IEEE ICC'12 Symposium on Communication and Information Systems Security* (2012).

[10] Dixit, S., Gupta, S., and Ravishankar, C. Lohit: An online detection & control system for cellular sms spam. In *IASTED Communication, Network, and Information Security* (2005).

[11] Gu, G., Perdisci, R., Zhang, J., and Lee, W. BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th USENIX Security Symposium (Security'08)* (2008).

[12] Jiang, N., Jin, Y., Skudlark, A., Hsu, W.-L., Jacobson, G., Prakasam, S., and Zhang, Z.-L. Isolating and analyzing fraud activities in a large cellular network via voice call graph analysis. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (New York, NY, USA, 2012), MobiSys '12, ACM, pp. 253–266.

[13] Krügel, C., Toth, T., and Kirda, E. Service specific anomaly detection for network intrusion detection. In *Proceedings of the 2002 ACM symposium on Applied computing* (New York, NY, USA, 2002), SAC '02, ACM, pp. 201–208.

[14] Lever, C., Antonakakis, M., Reaves, B., Traynor, P., and Lee., W. The core of the matter: Analyzing malicious traffic in cellular carriers. In *Proceedings of the ISOC Network & Distributed System Security Symposium (NDSS)* (2013).

[15] Liu, J., Ke, H., and Zhang, G. Real-time sms filtering system based on bm algorithm. In *International Conference on Management and Service Science (MASS), 2010* (2010).

[16] Lookout. Security alert: Spamsoldier. `http://goo.gl/t2oit`.

[17] Lookout. You are a winner! or are you? the walmart gift card scam. `http://goo.gl/WX6ps`.

[18] Security, L. M. Ggtracker technical tear down. `blog.lookout.com/wp-content/uploads/2011/06/GGTracker-Teardown\_Lookout-Mobile-Security.pdf`, 2011.

[19] Sekar, R., Gupta, A., Frullo, J., Shanbhag, T., Tiwari, A., Yang, H., and Zhou, S. Specification-based anomaly detection: a new approach for detecting network intrusions. In *Proceedings of the 9th ACM conference on Computer and communications security* (New York, NY, USA, 2002), CCS '02, ACM, pp. 265–274.

[20] Sommer, R., and Paxson, V. Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on* (may 2010).

[21] Thottan, M., Liu, G., and Ji, C. Anomaly detection approaches for communication networks. In *Algorithms for Next Generation Networks*, G. Cormode and M. Thottan, Eds., Computer Communications and Networks. Springer London, 2010, pp. 239–261.

[22] Wang, C., Zhang, Y., Chen, X., Liu, Z., Shi, L., Chen, G., Qiu, F., Ying, C., and Lu, W. A behavior-based sms antispam system. *IBM J. Res. Dev. 54* (November 2010).

[23] Wolda, H. Similarity indices, sample size and diversity. *Oecologia 50*, 3 (1981), 296–302.

[24] Yen, T.-F., and Reiter, M. K. Traffic aggregation for malware detection. In *DIMVA '08: Proceedings of the 5th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (2008).

[25] Zhou, Y., and Jiang, X. Dissecting android malware: Characterization and evolution. In *IEEE Symposium on Security and Privacy* (2012).