SherLock vs Moriarty: A Smartphone Dataset for Cybersecurity Research

Yisroel Mirsky
yisroel@post.bgu.ac.il

Asaf Shabtai
shabtaia@bgu.ac.il

Lior Rokach
liorkk@post.bgu.ac.il

Bracha Shapira
bshapira@bgu.ac.il

Yuval Elovici
elovici@bgu.ac.il

Department of Software and Information Systems Engineering
Ben-Gurion University, Beer Sheva

ABSTRACT

In this paper we describe and share with the research community, a significant smartphone dataset obtained from an ongoing long-term data collection experiment. The dataset currently contains 10 billion data records from 30 users collected over a period of 1.6 years and an additional 20 users for 6 months (totaling 50 active users currently participating in the experiment).

The experiment involves two smartphone agents: SherLock and Moriarty. SherLock collects a wide variety of software and sensor data at a high sample rate. Moriarty perpetrates various attacks on the user and logs its activities, thus providing labels for the SherLock dataset.

The primary purpose of the dataset is to help security professionals and academic researchers in developing innovative methods of implicitly detecting malicious behavior in smartphones. Specifically, from data obtainable without superuser (root) privileges. To demonstrate possible uses of the dataset, we perform a basic malware analysis and evaluate a method of continuous user authentication.

Keywords

Smartphone dataset; machine learning; malware; forensics; anomaly detection; continuous authentication.

1. INTRODUCTION

Today smartphones are ubiquitous, serving as a mobile mechanism for centralizing one's private information, accounts, contacts, and communication services. Given this, over the years smartphones have become an attractive target for cyber-attacks [1, 2]. Trojans are malicious applications that look legitimate. Users of Android devices do not always carefully review or understand app permissions during installation [3], enabling trojans to be installed and gain access to sensitive resources, even without exploiting vulnerabilities.

AISec'16, October 28 2016, Vienna, Austria

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4573-6/16/10…$15.00

DOI: http://dx.doi.org/10.1145/2996758.2996764

After infection, one approach for detecting a malicious behavior on smartphone is to perform dynamic analysis on the device itself [4–10]. However, many such *on-site* solutions require either root system privileges or changes to the kernel code. To root an Android device, the user needs to perform special steps that may require some technical background. Furthermore, rooted devices are exposed to many more vulnerabilities, thus causing security architectures to lose their effectiveness within these environments [11, 12]. Therefore, approaches that require root privileges are difficult to market widely via app stores. Moreover, methods that require changes to the kernel code are unlikely to reach the common user unless Google or smartphone manufacturers adopt these changes.

Given this distribution problem, it is clear that there is a need to develop security solutions that are based on low-privileged monitorable features: information sources that can be sampled without root privileges or changes to the smartphone's OS. Examples of these features are an application's memory consumption and the device's acceleration. In [13] the authors explore how different sets of monitorable features can be used to detect various malicious activities. Low-privileged monitorable features has been used for malware detection [14–16], data leakage detection [17, 18], continuous authentication [19–21], and in other situations. A great challenge in researching these domains is the lack of a public dataset – specifically a labeled smartphone dataset that captures ongoing attacks within the low-privileged monitorable features.

To fill this gap, we have established a long-term smartphone data collection experiment for cybersecurity research. We developed a data collection agent called SherLock and a malicious agent called Moriarty. SherLock collects a wide variety of low-privileged monitorable features at a high sample rate. In parallel, Moriarty perpetrates different cyber-attacks and leaves clues (time series event labels) for SherLock to collect. Both agents are currently deployed on 50 Galaxy S5 smartphones. The full data collection experiment began in February 2015 and is expected to continue until at least January 2018, for a total of three years. The SherLock dataset currently contains 10 billion data records and is now growing at a rate of approximately 670 million records (46 billion data points) a month. With the consent of the volunteers, a version of the dataset without explicit identifiers available online (at no charge) to academics and the research community.[1]

[1] `http://bigdata.ise.bgu.ac.il/sherlock/`

The SherLock dataset is unique with respect to existing public datasets for the following reasons:

Temporal Resolution. The SherLock dataset offers monitorable features sampled at a significantly higher frequency. Furthermore, the motion sensor data provides temporal coverage that is 80 times greater than other similar datasets. These sample rates reduce the phone's battery life to four hours with the original battery. To make the experiment practical for our volunteers, we provided each volunteer with a larger battery that extends the battery life to a full day.

The Collected Data. The SherLock dataset offers wider range of data. For example, we collect most of the Linux level memory, CPU, and scheduler information from every running application. Moreover, we also collect a greater range of aggregated statistics for the motion sensors.

Explicit Labels. The SherLock dataset offers labels that capture the moment various types of malware perform their malicious activities. We achieve this by running our own malware on the devices, where the behaviors are based on real malware samples that exist in the wild.

Therefore, our contributions in this paper are as follows:

- We publish and describe a massive smartphone dataset that contains a wide variety of low privileged monitorable features. Compared to existing public datasets, the SherLock dataset has the highest temporal resolution and is the only one that provides explicit labels for smartphone cyber-attacks captured on-site. This dataset can serve as a benchmark dataset for time series anomaly detection, data mining, forensics, and more. The dataset can also be used for fields other than security such as social sensing and recommender systems.
- We demonstrate how the dataset can be used to analyze malicious behaviors. As examples, we analyze two different malicious behaviors captured by sampled Linux-level features. We also demonstrate how this dataset can be used to develop and evaluate continuous user authentication algorithms. Utilizing the dataset, we found that a physical device theft can be detected in 15-105 seconds (2-8 observations) with a low false reject rate (FRR).

The remainder of the paper is organized as follows: In section 2 we provide the experiment setup and agent design. In section 3 we describe the dataset, provide statistics, and suggest possible uses. In section 4 we demonstrate two uses of the dataset: the detection of malware and the detection of device theft. In section 5 we compare our dataset to existing datasets, and in section 6 we summarize with a conclusion and future work.

2. EXPERIMENT SETUP

The objective of the data collection experiment is to provide security researchers access to a labeled dataset containing a wide variety of low-privileged monitorable smartphone features that capture both regular usage and cyber-attacks. Two smartphone agents were developed for this task: SherLock and Moriarty. Figure 1 presents an overview of the data collection experiment as described below.

SherLock is a data collection agent which collects data from a wide variety of low privileged monitorable features (hardware and software). The collected data is stored temporarily on the device and then batch uploaded to a server when WiFi connectivity is available. The server stores the data in a Hive database on a Hadoop cluster.

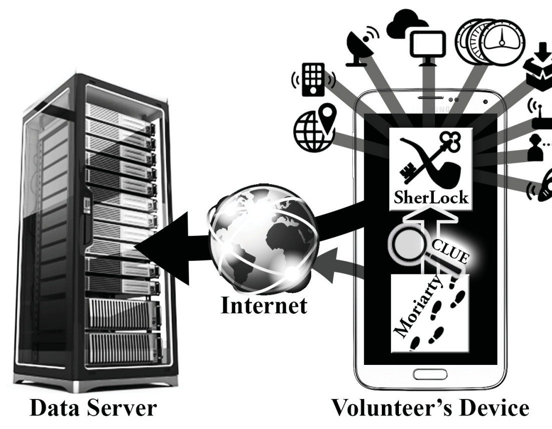


Figure 1: A conceptual overview of the data collection experiment. Moriarty performs malicious activities and leaves a log consisting of *clues* (entries). SherLock collects a wide range of data from low-privileged monitorable features, along with Moriarty's clues, and periodically uploads everything to a server.

Moriarty is a benign application paired with a malicious behavior. Both the benign application and its malicious behavior are changed every few weeks. Moriarty logs both the benign and malicious activities it performs. We refer to Moriarty's log entries as clues. Moriarty leaves clues on the device for SherLock to collect. The clues serve as explicit labels for the time series dataset collected by SherLock. In February 2015, 30 volunteers were provided with Galaxy S5 smartphones with SherLock installed. Then, one year later, 20 more volunteers joined the experiment and all 50 volunteers had Moriarty installed on their smartphones. None of the devices were rooted. The volunteers where required to use the provided device as their sole cellphone for at least two years. The volunteers were required to open the Moriarty application and use it at least once every few days for a few minutes (e.g., if Moriarty happened to be a game, then the volunteers would be required to play it). The volunteers were also asked to complete a survey. The survey

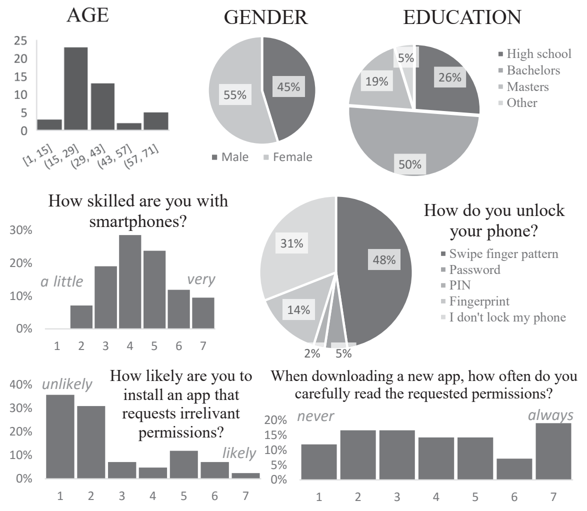


Figure 2: Sample results from the volunteer survey.

contained 118 questions about the volunteer's daily life, device usage, and security awareness. A sample of the results is presented in Figure 2. The survey results show that volunteers are diverse in their habits and experiences. The complete survey results are available online with the SherLock dataset.

The basic selection criteria for a volunteer was: (1) is between the ages 10 and 75, (2) has at least two years smartphone experience, (3) downloads on average at least one new app every two months, (4) has WiFi connectivity at home, and (5) uses a smartphone on a daily basis. From the candidates, we selected volunteers from at least three different geographic regions. All volunteers were given the same model phone (S5) to enable inter-user data analysis. In other words, so that we can capture the behavior of each malware under the context of different users using the same hardware as a baseline.

We will now elaborate on the two smartphone agents, following which we will present the version release timeline.

2.1 The Smartphone Agents

2.1.1 SherLock: Data Collection Agent

Framework. The SherLock data collection agent is based on the Google Funf framework [22]. The Funf Open Sensing Framework is an extensible sensing and data processing framework for mobile devices, developed by the MIT Media Lab. Funf was not designed for intense frequent feature monitoring, such as computing statistics on motion sensors. Therefore, we had to make modifications to the framework's processing pipeline in order to improve stability and reliability. Moreover, we added probes to the framework, such as a probe that collects statistics on all running applications. The source code for the SherLock agent, along with its modifications and its complete version history, is available online with the dataset.

In the Funf framework, monitorable features are referred to as *sensors*. Funf allows for developers to define groupings of sensors such that they will be sampled together at roughly the same time. These defined groupings are referred to as *probes*. A probe is activated (sampled) according to a configuration provided to the Funf's scheduler. For example, a probe that senses motion can be triggered once every five seconds or whenever some specific event occurs (e.g., the press of the home button). Sensors can retrieve data from either physical or virtual sources (e.g., external temperature or memory consumption). In general, there are two types of sensors: PUSH and PULL. PUSH sensors are event-based, such as sensing when an SMS arrives or when the screen is turned on. PULL sensors are collected periodically, such as by sampling the CPU utilization or the device's acceleration.

The data collected by SherLock is stored temporarily on the volunteer's device as a text file in JSON format. Once the file reaches 500MB, it is zipped to $\sim$50MB. Eventually, once the user connects to WiFi, all of the zip files temporarily stored on the device are uploaded to our server.

Probes. SherLock has seven probes for the PUSH sensors and five probes for the PULL sensors (described in Tables 8 and 9 in the appendix). Overall, there are many sensors divided amongst just a few probes. The probes were planned this way in order to *fuse* as many sensor records as possible to individual timestamps. This approach is advantageous when dealing with a large time series datasets because it minimizes the number of `JOIN` operations required on the final database when producing datasets. The most frequently sampled PULL probe is *T4*, sampled once every five seconds. Each time the *T4* probe is triggered, three software sensors are sampled. One sensor collects global features on the device's utilization and statistics down to the Linux level. The second collects statistics and Linux level features from each running application. The third collects features regarding the battery and power consumption. The Linux level features are obtained from the Linux virtual `/proc` directory which is accessible on unrooted Android phones. The information collected by *T4*'s three sensors can be found in Table 10 of the appendix.

Battery Consumption. The SherLock agent consumes a significant amount of battery power, because it samples many different sensors at a relatively high frequency. It is important that the devices have a normal battery life, otherwise the volunteers might leave their phones tethered to the wall charging for most of the day. This would affect the context and usage of the device captured in the collected data. To solve this issue, the volunteers were provided with a protective case that quadrupled the original battery capacity.

The fact that SherLock has high battery consumption does not mean that an algorithm developed with the SherLock dataset will be impractical. This is because:

1. The point of the SherLock experiment is to exhaustively collect as much information as possible in order to best identify the most effective features for different tasks. Therefore, algorithms based on the SherLock dataset will likely use a small subset of the sensors collected, and thus consume significantly less power.
2. Cost-aware data acquisition algorithms may be used to collect the same information more efficiently [23–25].
3. Motion sensors, which consumed much of the power, are becoming more efficient.

The last point has become evident with the growing popularity of wearable computing. For instance, the 2015 Freescale accelerometer (*MMA8452Q*) is approximately 63% more efficient than it was in 2012 (*MMA8450Q*)[2].

2.1.2 Moriarty: Malicious Behavior Agent

Framework. Moriarty is an agent that perpetrates attacks on the user's device while creating labels for the SherLock dataset. The attacks perpetrated by Moriarty are based on the following attack model: (1) A benign application is either initially given or repackaged to include additional code (spyware or some other malware). (2) The victim installs the app from a marketplace without realizing the consequences of the requested permissions. Given this model, we note that Moriarty does not need to exploit vulnerabilities on the device in order to perform its malicious activities.

In terms of its framework, Moriarty is a benign application to which a malicious behavior has been added, for example, a puzzle game that steals the user's contacts. Every few weeks Moriarty is updated to a new version (app + behavior) via the Google Play store. If the user forgets to use the Moriarty application for three days, then SherLock reminds the user to do so. For ethical reasons, each version of Moriarty does not contain actual malware code from the wild but rather contains a behavioral copy of a malware sample found in the wild. In order to protect the volunteer's privacy, all data sent by Moriarty over the Internet is scrambled (corrupted) before transmission.

[2]`http://www.nxp.com/files/sensors/doc/data_sheet/MMA8450Q.pdf` *and* `MMA8452Q.pdf`

Release in 2016	Version	Benign Application	Malicious Behavior	Description	Malware Sample
Q1	1	Puzzle Game	Contacts Theft	*Steals, encrypts, and transmits all contacts stored on the device.*	SaveMe/ SocialPath
	2	Web Browser	Spyware	*Either: 1) spies on location and audio, or 2) spies on web traffic and web history.*	Code4hk/ xRAT
	3	Utiliz. Widget	Photo Theft	*Steals photos that are taken and in storage, and takes candid photos of the user.*	Photsy/ Phopsy
Q2	4	Sports App	SMS Bank Thief	*Captures and reports immediately on SMSs that contain codes and various keywords. Volunteers periodically enter one of our websites, enter their phone# and then enter a code which we respond to them via SMS.*	*Spy. Agent.SI*
	5	Angry Birds	Phishing	*Makes fake shortcuts and notifications to login to Facebook, Gmail, and Skype.*	Xbot
	6	Game	Adware	*Gathers information and places ads, popups and banners.*	
Q3	7	Game	Madware	*Gathers private information and places shortcuts, notifications, and attempts to install new applications.*	
	8	Lock Screen	Ransom-ware	*Performs either: 1) lock screen ransomware, or 2) crypto ransomware.*	Simplocker.A/ SLocker
	9	File Manager	Click-jacking	*Tricks the user to activate accessibility services to then hijack the user interface.*	Shedun (GhostPush)
Q4	10	*None*	Device Theft	*The volunteer has a friend 'steal' his/her device in various ways, and records the moment when.*	
	11	Music Player	Botnet	*Either performs: 1) DDoS attacks on command, or 2) SMS botnet activities*	Tascudap.A/ Nitmo.A…
	12	Web Media Player	Recon. Infiltration	*Maps the connected local network and searches for files and vulnerabilities.*	

Table 1: The Moriarty releases for 2016

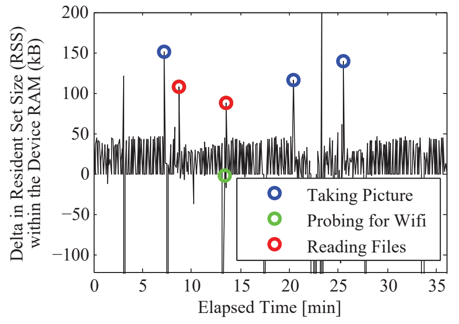


Figure 3: Changes in Moriarty `v3`'s memory consumption, as indicated by the recorded clues.

Table 1 presents the versions of Moriarty that have been released so far. The table also lists the malwares taken from the wild on which each version is based. In total, there will be at least 12 versions of Moriarty by the end of 2016. Note that `v10` does not follow the attack model since it is an experiment platform for capturing device theft behaviors. In that version the volunteer has a friend "steal" and use the in various manners. The labels recorded by Moriarty in this version indicate the moment the device was stolen.

Clues. The explicit labels Moriarty creates for the SherLock dataset are referred to as clues. Moriarty creates a clue for each significant benign and malicious action it performs. A benign action is one which the application does as part of its regular usage (e.g., a game makes a new view).

While Moriarty is running, it can be in one of two modes: malicious or benign. While in malicious mode Moriarty performs its malicious activities in parallel with its benign activities. In benign mode, Moriarty only performs benign activities. The purpose of the benign mode is to provide clean (normal) data that can be separated and used as a comparative dataset during the development of anomaly detection algorithms. If Moriarty's malicious behavior runs as a background service, the mode changes every 24 hours; otherwise it toggles each time the user runs the application.

A session is the duration that Moriarty is in benign mode or malicious mode. All clues recorded during a session are given a session ID. The session ID can be used to extract the desired mode or behavior from the time series data. In cases where Moriarty has two malware behaviors (i.e., `v2`,`v8` and `v11`) each behavior is performed in different sessions (i.e., they do not overlap in time).

In Figure 4 examples of clues from Moriarty `v3` are displayed. In this version, Moriarty was a widget that displayed the current CPU and memory utilization to the user while covertly taking front-facing photos, stealing photos from storage, and sending them offsite. The trace of the three clues belongs to a single session that is malicious. Note that in the first clue the `SessionType` is malicious, but the `Action` itself is benign (i.e., part of the application's legitimate behavior). The Details field elaborates on the action taken and sometimes includes specifics such as the traffic volume, duration, message content, and so on. All of these fields are parsed into the final SherLock dataset.

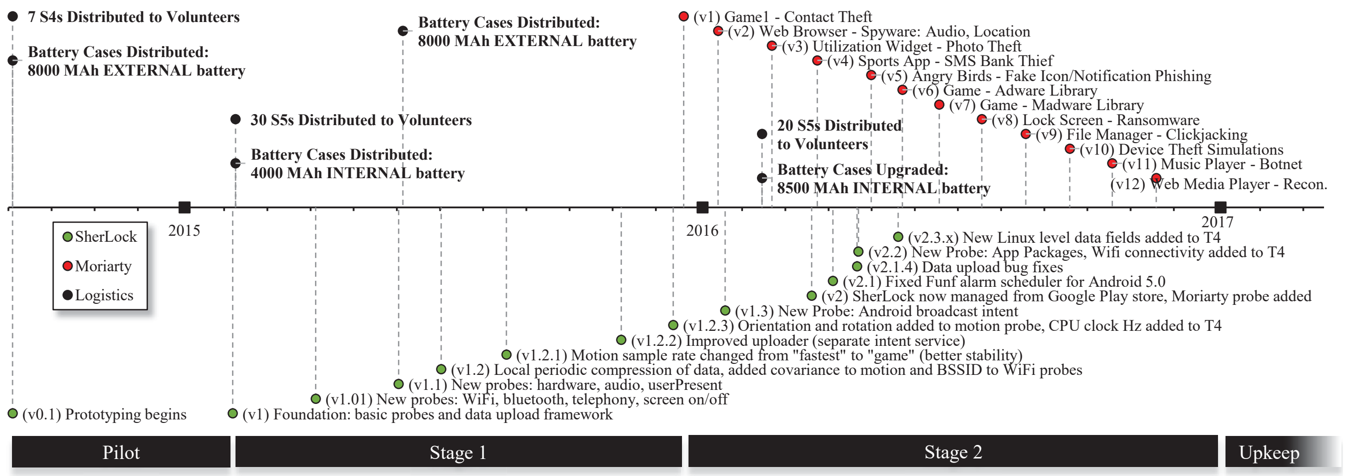
```
"Action": "App Mode Change",
"ActionType": "benign",
"Details": "App entered onCreate()",
"TimestampMili": 1453824930840,
"SessionType": "malicious",
"Version": "3.0",
"SessionID": 1

"Action": "Stealing photos",
"ActionType": "malicious",
"Details": "Found new photo from the gallery",
"TimestampMili": 1453825161930,
"SessionType": "malicious",
"Version": "3.0",
"SessionID": 1

"Action": "Sending photo",
"ActionType": "malicious",
"Details": "Sent photo successfully
 (duration [msec],size [bytes]);18640;4154704",
"TimestampMili": 1453825180575,
"SessionType": "malicious",
"Version": "3.0",
"SessionID": 1
```

Figure 4: Three examples of clues left by Moriarty `v3`.

Moriarty's activities are captured by the wide variety of sensors that SherLock samples. The clues help explain the captured data. As an example, Figure 3 plots the changes in Moriarty's resident set size (RSS) memory consumption, and the timestamps of Moriarty's clues.

2.2 Version Release Timeline

The data collection experiment was divided into four phases: Pilot, Stage 1, Stage 2, and Upkeep (Figure 5). In the Pilot phase, seven volunteers were provided with Galaxy S4 devices and initial prototypes were distributed and iteratively improved. In Stage 1, 30 volunteers were provided with a stable version of SherLock and a Galaxy S5 smartphone. At first, the volunteers were provided with a battery case that extended the internal battery, and a few months later they were provided with additional battery cases that charge the internal battery from an external battery. These technical differences are important to note, because the latter case charges the phone's internal battery as if it were plugged in to the wall, and thus affects the power consumption statistics.



7 S4s Distributed to Volunteers
Battery Cases Distributed: 8000 MAh EXTERNAL battery
30 S5s Distributed to Volunteers
Battery Cases Distributed: 4000 MAh INTERNAL battery
Battery Cases Distributed: 8000 MAh EXTERNAL battery
(v1) Game1 - Contact Theft
(v2) Web Browser - Spyware: Audio, Location
(v3) Utilization Widget - Photo Theft
(v4) Sports App - SMS Bank Thief
(v5) Angry Birds - Fake Icon/Notification Phishing
(v6) Game - Adware Library
(v7) Game - Madware Library
(v8) Lock Screen - Ransomware
(v9) File Manager - Clickjacking
(v10) Device Theft Simulations
(v11) Music Player - Botnet
(v12) Web Media Player - Recon.
20 S5s Distributed to Volunteers
Battery Cases Upgraded: 8500 MAh INTERNAL battery
2015
2016
2017
SherLock
Moriarty
Logistics
(v2.3.x) New Linux level data fields added to T4
(v2.2) New Probe: App Packages, Wifi connectivity added to T4
(v2.1.4) Data upload bug fixes
(v2.1) Fixed Funf alarm scheduler for Android 5.0
(v2) SherLock now managed from Google Play store, Moriarty probe added
(v1.3) New Probe: Android broadcast intent
(v1.2.3) Orientation and rotation added to motion probe, CPU clock Hz added to T4
(v1.2.2) Improved uploader (separate intent service)
(v1.2.1) Motion sample rate changed from "fastest" to "game" (better stability)
(v1.2) Local periodic compression of data, added covariance to motion and BSSID to WiFi probes
(v1.1) New probes: hardware, audio, userPresent
(v1.01) New probes: WiFi, bluetooth, telephony, screen on/off
(v0.1) Prototyping begins
(v1) Foundation: basic probes and data upload framework
Pilot
Stage 1
Stage 2
Upkeep

Figure 5: A timeline that summarizes the phases, major events, and agent version releases during the data collection experiment.

In Stage 2, Moriarty was deployed and 20 additional volunteers were added (for a total of 50 Galaxy S5 devices). Both agents were then managed through the Google Play store as opposed to manual APK updates. At that point all volunteers were supplied with a higher quality battery case that extends the internal battery (Zerolemon 8500MAh). In the Upkeep phase, further updates to SherLock and Moriarty will be sparse, but the data collection will continue until at least January 2018 (for a total of three years of data collection).

During Stages 1 and 2, SherLock was updated several times. These updates were performed to fix bugs, improve stability, and add new probes/sensors. A detailed version history for both Moriarty and SherLock is available with the dataset.

3. THE SHERLOCK DATASET

In this section the contents of the SherLock dataset are described in detail. We also identify uses of the dataset for the development of security solutions, as well as other research possibilities.

3.1 Description of the Dataset

The dataset currently contains approximately 600 billion data points in 10 billion data records. The dataset is currently growing at a rate of approximately 670 million records a month. To manage the *big data*, the dataset is stored on a Hadoop cluster as a Hive database. The volunteers were given the option to leave the experiment at any time, however only two volunteers left the experiment since it has begun. Together, the long-term volunteer participation and the high sensor sample rates make this dataset unique. A full comparison to other existing datasets is presented later in section 5.

The SherLock dataset is organized into data tables, one for each SherLock probe (Tables 8 and 9). Some of the sensors belonging to these probes return a variable number of records when they are sampled – specifically, the *WiFi*, *Bluetooth*, and *Local App Stats* sensors, since they perform surveys over a variable number of entities. The data records from these sensors cannot be stored in the data tables of their parent probes. Therefore the data from these sensors are stored in separate tables.

Every data table has the fields `uuid`, `userid`, and `version`. `uuid` is the Unix millisecond timestamp of when the record was collected. `userid` is a unique identifier for the volunteer to whom the record belongs. Finally, `version` is the agent's software release code (see Figure 5). The SherLock's 15 data tables with their current number of records are listed in Table 2.

	Data Table	Number of Records
PUSH	*Call Log*	443,175
	SMS Log	245,693
	Screen Status	2,608,766
	User Presence	685,910
	Broadcast Intents	95,471,166
	App Packages	108,612
	Moriarty	650,625
PULL	*T0*	242,762
	T1	14,050,156
	WiFi	54,654,980
	Bluetooth	2,945,238
	T2	43,383,170
	T3	85,861,126
	T4	180,012,794
	Application	9,271,351,994
	Total:	9,752,716,167

Table 2: The data tables and number of data records in the SherLock dataset as of August 2016.

3.2 Ethical Concerns

The SherLock data collection experiment has passed Ben-Gurion University's Ethics Committee. To protect the privacy of our volunteers, several concerns were addressed.

Data Transmissions. In order to protect the privacy of our volunteers, all data transfers from the SherLock agent to our servers are encrypted and all data sent by the Moriarty malicious agent is scrambled before transmission.

Identifiers. We have received permission from the volunteers to allow others to use their data for research, provided that we anonymize their data. Therefore, network identifiers such as SSIDs, cell tower IDs, and MAC addresses have been hashed. Moreover, the phone numbers in the SMS and call logs have been hashed as well.

Geolocation. The geolocation of each volunteer has been anonymized by performing k-means clustering over the volunteer's longitude, latitude, and altitude. This approach was proposed in [26] and is a method that achieves k-anonymity

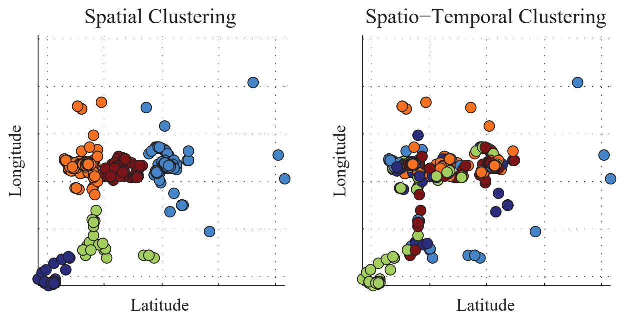


Figure 6: A user's longitude and latitude after k-means clustering ($k = 5$) when using only the spatial features (left), and also the cyclic day of week features (right).

via generalization. This means that those who receive the dataset will have the volunteer's location in terms of cluster IDs. In order to provide various degrees of granularity, we repeated the clustering for $k = \{5,10,25,50,75,100\}$. Time has a contextual importance to a user's geolocation. For example, being at work on the weekend is a different situation than being at work during the week. To provide this spatio-temporal information in an anonymous manner, we provide additional clusterings using both geolocation and temporal features with $k = \{5,25,100\}$. Specifically, the temporal features we consider are cyclic time of day, and week [27]. Cyclic time is defined as,

$$t_c \equiv \left(\cos\left(\frac{2\pi t}{T}\right), \sin\left(\frac{2\pi t}{T}\right)\right) \tag{1}$$

where T is some cycle of time, and t is a point in that cycle. For example, when considering a daily cycle, we would take $T = 24$ such that the time 11:39pm would be interpreted as $t = 23.65$. Thus, t would be interpreted in cyclic time as $t_c = (0.9958, -0.0915)$. The advantage of cyclic time is that it captures temporal contexts better than a single scalar value. This is especially true when considering Euclidean distance between the hours $t_1 = 0.01$ and $t_2 = 23.9$. In this case, $d_{\text{euclid}}(t_{(c,1)}, t_{(c,2)}) \ll d_{\text{euclid}}(t_1, t_2)$. Figure 6 shows the difference in clustering spatial features with and without the cyclic temporal features. In summary, by providing the volunteer's location via spatial and spatio-temporal cluster IDs, we provide enough contextual information to make inferences using the collected data without compromising the volunteer's privacy.

3.3 Dataset Usages

The following are some possible cybersecurity usages for the dataset:

Malware Detection & App Profiling. The dataset can be used to detect malware and profile applications via implicit application activity (network traffic, CPU/memory utilization, etc.). A survey which maps the implicit detection of malware to monitorable features smartphones is available in [13]. Moreover, the dataset has many contextual features (such as the device's location, motion, and battery consumption) that can be used to improve the recognition of malicious intents.

With regard to malware samples in the dataset, researchers have two options: (1) each version of Moriarty can be analyzed as a different type of malicious application (i.e., spyware, adware, etc.), or (2) the hash values of installed applications' APKs can be used to detect real malware on the volunteer's devices. The hash values are provided by the App Package probe whenever the application is installed or updated, and VirusTotal[3] can be used to map the hashes to known malware. We found that 3.5% of applications on our volunteers' devices had a virus signature, and ∼1% had at least three positive detections from different anti-virus solutions.

Malware Analysis. With the APK hashes and version of Moriarty, it is possible to use the dataset to analyze the impact different malwares have on the device. Specifically, it is possible to see what low-privileged monitorable features are affected (and when) by the malware's activities. This analysis may be used to help develop an efficient way of detecting malicious behaviors from low-privileged monitorable features, obtainable without root privileges.

Continuous User Authentication. The dataset can be used to develop continuous authentication algorithms. This is because each volunteer participates in the experiment for over a year, an thus the data presents the inherent challenges: concept drifts, behavioral noise, and false positives when the device is shared with others. Researchers can use the dataset for this task in one of two ways: (1) by using the labeled data from Moriarty `v8` in which the volunteers participate in simulated device thefts, or (2) by slicing a volunteer's data at one point, and then attaching another volunteer's data as the continuation.

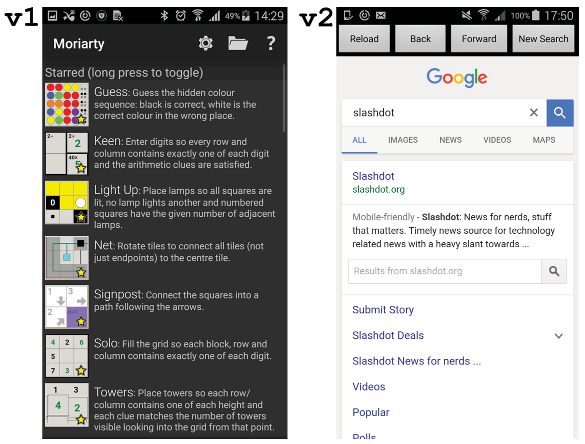
Context-based Security. The dataset can be used to derive the general context of the user and his/her device, as well as the specific contexts in which events are performed. These contexts can be used to implement and evaluate different context-based security mechanisms. For example, data leakage can be detected by analyzing the context in which information is sent from the device. For example, an SMS that is sent while the screen is off, the device is motionless, or while the user is driving is likely to be the result of and automated act of malware. Moreover, the contexts may also be extracted as a rule-set as a means of improving resource access control (i.e., only allowing applications to access certain resources under acceptable pre-defined contexts).

Security Related Statistics. The dataset can be used to derive security-related statistics. For example, a score that reflects a user's security awareness can be computed based on the user's WiFi usage patterns, installed applications (possibly malware) and their permissions, and the user's general behavior. Each volunteer completed a survey which included 45 security-awareness questions. The survey responses can be used to confirm the calculation of these scores or contribute to their calculation.

Feature Monitoring & Extraction. The dataset contains data sampled at a relatively fast rate (with respect to the limitations of the modern smartphone). Thus, researchers who are investigating security algorithms can also consider the trade-off between sample rates (more power consumption) and accuracy. Furthermore, feature selection can be performed to reveal the best features for a particular problem, or to provide an analysis of an application or user's behavior.

We note that this dataset can be used for research in domains that are not strictly security related. For example, context aware recommender systems, event prediction, user personalization and awareness, location prediction, and more. The

[3]`https://virustotal.com/`



Guess: Guess the hidden colour sequence: black is correct, white is the correct colour in the wrong place.

Keen: Enter digits so every row and column contains exactly one of each digit and the arithmetic clues are satisfied.

Light Up: Place lamps so all squares are lit, no lamp lights another and numbered squares have the given number of adjacent lamps.

Net: Rotate tiles to connect all tiles (not just endpoints) to the centre tile.

Signpost: Connect the squares into a path following the arrows.

Solo: Fill the grid so each block, row and column contains exactly one of each digit.

Towers: Place towers so each row/column contains one of each height and each clue matches the number of towers visible looking into the grid from that point.

Mobile-friendly - **Slashdot**: News for nerds, stuff that matters. Timely news source for technology related news with a heavy slant towards ...

Figure 7: Screenshots of Moriarty. Left: `v1`, puzzle games. Right: `v2`, a web browser.

dataset also offers opportunities that aren't available in other datasets. For example, the dataset contains the SSID and signal strength of the connected WiFi access point (AP) which is sampled once every 5 seconds by the *T4* probe and is captured every second or less by the *Broadcast Intents* probe. This data can be used to infer the volunteer's movement and activities while indoors or as contextual information for some other task. A full comparison between the data collected in the SherLock dataset and other publicly available datasets is discussed later in section 5.

It is important to note that neither root privileges nor changes to the Android OS are required to monitor the features found in the SherLock dataset (as of Android 5.0). Therefore, solutions based on this data are applicable and accessible to a large portion of the Android device population.

4. PRELIMINARY RESULTS

4.1 Malicious Behavior Analysis

4.1.1 Overview

Intuitively, each type of malware affects the device's resources differently. In [13], the authors present a mapping between types of malwares and the sensors that can be used to detect them. However, without a real world dataset to compare to, some features and their importance may be overlooked. In this section, we examine the usage of the SherLock dataset as a means for analyzing the impact malicious code has on low-privileged sensors. Specifically, we focus on the system's resource and battery sensors.

In this analysis we examine two versions of Moriarty separately: versions 1 and 2. In `v1`, Moriarty was a puzzle game that would periodically steal the user's contacts and transmit them to a web server. In `v2`, Moriarty was a web browser that, depending on the session, would send information to a web server, either: (1) the user's location every 60 seconds, recorded audio from the entire session, and every accessed URL (upon clicking), or (2) the user's contacts, bookmarks, device accounts, and the last two months of web history after 30 seconds. Figure 7 presents screenshots from Moriarty `v1` and `v2`.

We remind the reader that Moriarty has benign sessions during which all activities performed by the Moriarty application (e.g., the web browser in `v2`) are benign (with no added malicious behavior).

4.1.2 Evaluation Setup

One dataset was created for each version. Each dataset represents the resource utilization of Moriarty and the entire device at the moment of each of Moriarty's malicious and benign actions. The dataset for Moriarty `v1` was made in the following way: For each volunteer, the volunteer's `v1` clues were extracted. Then, for each clue, the instance $\langle \vec{x}, y \rangle$ was added to the dataset. $\vec{x}$ is a vector containing the numeric values of the *T4* probe that were sampled closest to the clue's timestamp, $y \in \{0,1\}$ is a nominal label indicating whether the clue's ActionType was malicious or benign. The instances were obtained by performing the relational `JOIN` between the *Application* and *T4* data tables over the `uuid` field (collection timestamp), where the `application_name = 'Moriarty'` and `version = '1.0'`. The same process was repeated to create the dataset for `v2`.

In the end, each observation $\vec{x}$ contained 57 features. Some of the features from *T4* were not included, since they were added to SherLock after Moriarty `v2` (see the Figure 5).

4.1.3 Evaluation Results

To determine the impact Moriarty `v1` and `v2` have on the features, we examine *each* feature's dcorrelation and information gain (IG) to the labels. Features with a higher correlation than others are consistently affected by the malicious actions. Similarly, a feature with a higher IG indicates that the feature is better at discriminating between the benign and malicious actions [28].

Figure 8 lists the features used in datasets `v1` and `v2` with their IG (left) and correlation (right) to the label. The IGs of the traffic features in the `v1` dataset are not a surprise, since the game (the benign behavior) does not use the Internet. However, it is interesting to note the `battery_temperature` has a correlation, even greater than the `battery_voltage` or `battery_plugged features`. This is a good indication that although the SherLock agent affects battery consumption, the *battery consumption* data in the SherLock dataset can still be used to model app behaviors. Finally, the `otherSharedDirty` memory feature has a low correlation but a high information gain. This is because the dirty memory levels are noisy with respect to the malicious and benign activities. However, a large increase is a strong indication that something malicious has likely occurred.

With regard to the `v2` dataset, `importance` has both a high correlation and IG and is therefore strongly affected by the malware. The `importance` feature is an Android property that indicates the relative importance given by the system to the application [29]. This strong relationship is likely due to the spyware's activities relating to location tracking and audio recording. Thus, `importance` alone may be a good feature for detecting spyware activities.

Lastly, we note that for both datasets, the private and shared memory features ranked consistently higher than the others. This is a good indication that malicious behaviors in general can be detected implicitly by monitoring the various memory utilization features. This is an interesting prospect, and we encourage further research in this domain.

4.2 Continuous Authentication

4.2.1 Overview

One in ten smartphone owners in the US are victims of phone theft [30], and 36% of owners do not secure their devices [31]. One way to passively secure a smartphone is to use continuous authentication (CA). CA is the process of

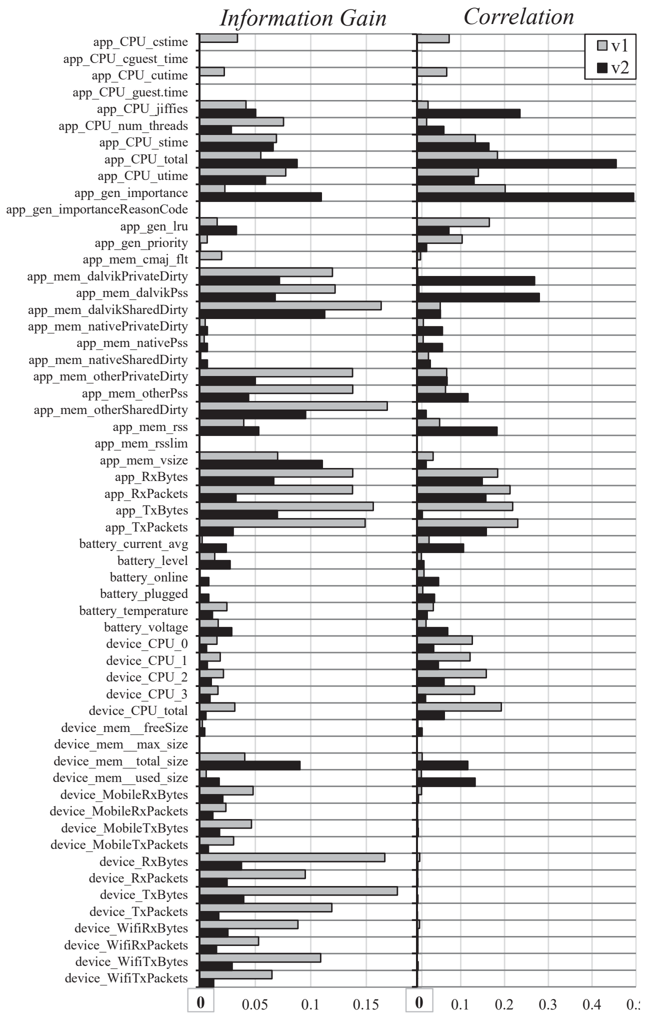


Figure 8: The information gain and correlation of the features to the labels in the `v1` and `v2` datasets.

continuously examining the current user's behavior and then subsequently de-authenticating the user when suspicion arises (e.g., by locking the device) [32].

Some of the inherent challenges of developing CA algorithms are as follows: (1) the sensor stream is very long and possibly unbounded, (2) the user's behavior is noisy, and 3) concept drifts and gradual changes in the data's distribution inevitably exist [33]. The SherLock dataset contains motion sensor data that has been continuously sampled from tens of users for over a year. Therefore, the dataset captures the challenges listed above, and is therefore a good resource for the development and evaluation of CA algorithms.

4.2.2 Evaluation Setup

Consider the following attack scenario. The attacker does not regularly handle a certain user's smartphone. The attacker wants to either steal or access an application on the user's smartphone. To do so, the attacker waits until the phone is left unattended on a table and then picks it up and begins to use it. In this scenario we assume that the attacker is familiar with the device and its interface.

To demonstrate the value of the SherLock dataset, we have developed a simple CA algorithm for the above scenario and evaluated it using the dataset. For this evaluation we simulated device thefts from stationary surfaces (e.g., a table top) and examined the device motion patterns from the *T3* probe. Specifically, we extracted three motion features from the acceleration, harmonics of the acceleration, and the gyroscope. Table 3 summarizes the features taken. The magnitudes were calculated as $\sqrt{x^2+y^2+z^2}$ where x, y, z are the values taken from the respective axis.

Feature	Description	Units
$\lVert\bar{a}\rVert$	Magnitude of average acceleration	m/sec^2
$\lVert\bar{g}\rVert$	Magnitude of average gyroscopic rotation	rad/sec
$\lVert FFT_1(a)\rVert$	Magnitude of strongest harmonics	Hz

Table 3: The features extracted from the SherLock dataset for the continuous authentication dataset.

The datasets for the evaluation were created from the SherLock dataset in the following way. Let user-A be the user we are trying to protect and user-B be the thief. A training set was formed with a 31 day data stream of user-A's motion data (180,000 records).

The test set was formed by: (1) Taking a segment consisting of user-A's next 20,000+ observations, up until the moment that the device was placed stationary on a flat surface (e.g., a table) for a period of time. (2) Taking a segment of 30,000 observations from user-B's data, from the moment that user-B lifted up his/her phone from a stationary surface. (3) Joining user-A's segment with user-B's segment to form a single nine day data stream. This process was repeated for several different train and test users.

The continuous authentication algorithm we used is a three-step process where (1) the extracted motion features are fed one-by-one into a stream clustering algorithm, (2) the sequence of outputted cluster IDs are applied to a Markov chain to compute transition probabilities, and (3) the anomaly scores are computed with the outputted transition probabilities.

For the stream clustering algorithm, we used pcStream, since it is effective at dynamically detecting situations in data streams while accounting for concept drift [34]. The input of pcStream is an unbounded numerical data stream, and the output is a sequence of nominal labels, each indicating the respective situation. A situation can be viewed as a distribution, or cluster, in geometric space that may overlap with other situations.

For the Markov chain, we used an extensible version that can be incrementally updated [35]. The model captures the temporal behavior of the outputted sequence of labels, i.e., the transitions between the situations exhibited by the stream.

For anomaly scoring, we computed the average transitional probability over a sliding window of length k. More formally, let M be the Markov chain trained on a single user's data stream where $M[s_i,s_j]$ is the transitional probability between the situation s_i and s_j, and let x_t be the present assigned situation at time t. Let $S_t = x_{(t-k)},\ldots,x_t$ be the sequence of the last k observed clusters until time t. Finally, the anomaly score of S_t is calculated by

$$score_M(S_t) = \frac{1}{k-1}\sum_{i=1}^{k-1} M(S_t[i],S_t[i+1]) \qquad (2)$$

4.2.3 Evaluation Results

In order for this evaluation to be fair, all of the phones must be the same hardware/model. The Galaxy S5 phone series has

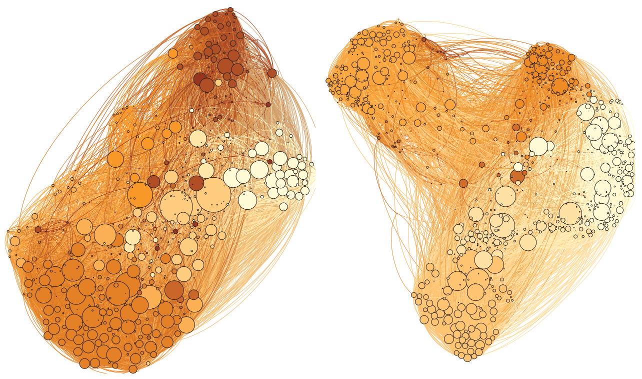


Figure 9: The Markov chains of Users 1 and 8.

	FRR	FRs Over Test Sets		Average Detection Delay	
User	**Mean**	**Mean**	**Std.**	**Minutes**	**Samples**
1	7.86E-05	1.571	2.225	1.679	7.716
2	4.29E-05	0.857	1.069	1.821	8.284
3	2.29E-04	4.571	9.624	1.643	7.572
4	4.29E-05	0.857	1.574	1.179	5.716
5	0	0	0	0.25	2
6	4.36E-04	8.714	11.161	6.643	27.572
7	2.81E-03	56.143	101.570	3.143	13.572
8	0	0	0	0.25	2

Table 4: Results from the continuous authentication experiment using the SherLock dataset.

different models such as those with either quad or octa-core CPUs. Therefore, for this evaluation, we only used the data of eight volunteers, all having the exact same model of S5.

Figure 9 plots two of the users' Markov chains using the graph visualization tool Gephi [36]. In this figure, each vertex is a pcStream situation (state in the Markov chain), and an edge indicates an observed transition. The size of a vertex indicates the context's in-degree (popularity), the color of a vertex indicates the context's community using the algorithm from [37], and the color of an edge indicates the transition's probability (darker is more probable). The figure illustrates that each user has distinct situational behaviors as captured by their locomotion.

The results of using (2) on the device theft datasets are presented in Table 4. The table shows that the algorithm has a low false reject rate (FRR) with an average detection delay of two minutes (eight observations). Faster sampling rates may improve these results. The proposed algorithm is only a demonstration of how the SherLock dataset can be used for cybersecurity research. We encourage researchers to use the SherLock dataset to improve results further.

5. RELATED WORKS

In the past, smartphone datasets have been collected for research in a variety of domains. For example, activity recognition, reality mining, and security. These datasets vary in the scope of the data collected and the duration of the experiment. In most cases the datasets were not long-term (i.e., contained less than three months of data per participant), were crafted with a specific domain problem in mind, and were not publicly released online.

In Table 5 we present a comparison between the existing long-term smartphone datasets that have been made publicly available. From Table 5 it is clear that the SherLock dataset rivals the other datasets with respect to the variety and sample rates. The Device Analyzer (DA) dataset rivals the SherLock dataset with regard to the number of users, however, the SherLock dataset exceeds the DA dataset in the following aspects: *Temporal Resolution*, *Motion & Application Sensors*, and *Explicit Malware Activity Labels*. We will now compare each of these aspects in detail.

5.1 Temporal Resolution

In the DA dataset, the PULL sensors are probed once every five minutes. In comparison with the SherLock dataset, the PULL sensor with the lowest sample rate (that does not sample configurations) is *T1* at once every minute, and the highest is *T4* at once every five seconds. We also note that we collect the current WiFi signal strength (`RSSI_CHANGED`) via the *Broadcast Intent* probe approximately every second although it is a PUSH sensor. A complete comparison is available in Table 6.

With regard to the application sensors, in the DA dataset the five minute interval is too long to catch malicious behaviors. This is because these behaviors can occur within seconds. Furthermore, information can be totally lost from the application probe, since a user can open and close an application within that window of time. In the SherLock dataset we sample the application statistics every five seconds.

We define temporal coverage as the percent of time that a sensor is being sampled at the highest frequency. The DA dataset samples the motion sensors for a duration of one second every five minutes. Therefore, the temporal coverage of DA's motion sensor data is ∼0.003%, compared to ∼26.666% for SherLock. Table 7 presents a comparison between the datasets' temporal resolutions.

The DA's motion sensor coverage is too low to be properly used as a contextual feature for an application's behavior (e.g., detecting data leakage by modeling the accompanying device motion). Moreover, security solutions that consider a device's motion require considerably large temporal coverage. For example, in continuous authentication an attacker can do a significant amount of damage (in terms of data theft) within the five minute gap of time between samples.

The DA dataset does include intervals in which sensors are sampled at a high frequency (2 Hz). During these intervals he following are sampled: the screen lock state, last started app, global network traffic stats, and CPU time in states for each core. However, these intervals are activated randomly only 10% of the time, only while the screen is on, and are no longer than five minutes in duration. Therefore, this data is too sporadic and too general to accurately capture both the user and applications' behaviors.

5.2 Motion & Application Sensors

The SherLock dataset has more motion and application related information than the DA dataset. For the motion sensors, DA only computes the mean and standard deviation of the captured samples. In addition to these features, SherLock provides the median, middle sample, covariance between axes and FFT statistics. The FFT is particularly useful, since it has been shown to improve inference based on activity recognition [43].

DA's application statistics do not provide a large number of Linux level features. For example, the state (sleep, zombie, etc.), number of active threads, scheduler priority, and various memory features. This information is highly useful for observing an application or service's behavior. In order

Dataset	*Objective*	*# of users*	*Devices*	*Launch*	*Duration*	*Feature*	*Sample Rates*
SherLock	Implicit cybersecurity & general mobile phone data collection	50 (national)	Samsung Galaxy S5	2014	ongoing	Device settings/mode, running applications, audio features, BT/WiFi probe, network stats, telephony, location, power, screen on/off, motion, call/SMS log, Moriarty activity, broadcast intents, IO interrupt counts, others.	Fastest: every 5 seconds Most data: every minute
Device Analyzer	General mobile phone data collection	30,443 (international)	Heterogeneous: Android	2011	ongoing	Device settings/mode, running applications, audio features, BT/WiFi probe, network stats, telephony, location, power, screen on/off, motion, call/SMS log, others.	Every 5 minutes
LiveLab Project	Measuring wireless networks and smartphone users	34 (students)	iPhone 3GS	2010	1 year	App, WiFi, cellular, device activity, call/SMS logs, battery, network traffic, others.	Every 15 minutes
LDCC	Social sensing	170 (locals)	Nokia N95	2009	2 years	Location, call/SMS log, BT/WiFi probe, telephony, motion, audio, app events, calendar entries, phonebook entries	Every 60-300 seconds
Social Evolution	Social sensing	80 (students)	unknown	2008	1.75 years	BT and WiFi signal strength, location labels, call/SMS logs.	Every 6 minutes
Reality Mining	Social sensing	100 (faculty and students)	Nokia 6600	2004	9 months	Mobile and BT signal strengths, location, call/SMS logs, running Nokia applications.	Every 6 minutes

Table 5: Existing long-term smartphone datasets that are publicly available: SherLock, Device Analyzer [38], Live-Lab [39], LDCC [40], Social Evolution [41], and Reality Mining [42].

Sensors	**Sample rate:** *every...* *SherLock*	*DA*
Location, Cellular, WiFi, Bluetooth, and Device Status	T1: 5 seconds	300 seconds
Motion Sensors	T2: 10 seconds	
Light & Audio Sensors	T3: 15 seconds	
Applications & Resource Utilization	T4: 60 seconds	

Table 6: A comparison between the SherLock and Device Analyzer dataset PULL sensor sample rates.

	SherLock	**Device Analyzer**	
Application Statistics	Every 5 sec. (0.2 Hz)	Every 5 min. (0.003 Hz)	*Probe Frequency*
Motion Sensors	Every 15 sec. (0.067 Hz)		
	4 seconds	1 second	*Probe Duration*
	26.666%	0.003%	*Temporal Coverage*

Table 7: A comparison between the application and motion sensor temporal resolutions in the Device Analyzer and SherLock datasets.

to provide a full comparison, we have underlined the fields in Table 10 of the appendix which are available in the DA dataset. Overall, SherLock collects over five times more data points from each application than DA. Moreover, we collect useful global device information such as various IO interrupt counts and all Android broadcast intents that occur.

In summary, the SherLock dataset provides both a higher temporal resolution and a larger scope of sensors than the DA dataset. The main reason for this is probably because DA's volunteers are not provided with extended batteries and receive no compensation for participation.

5.3 Explicit Malware Activity Labels

The SherLock dataset provides explicit event labels that capture the activities of malware on the device (Moriarty's clues). To the best of our knowledge, there are no other published smartphone datasets which offer explicit labels that capture a wide variety of malicious behaviors. Moreover, we include the hash of each installed application's APK. This is valuable information, since the hash can be cross-referenced with VirusTotal's database in order to detect malwares that may exist on volunteers' devices. This extends the set of labeled instances in the SherLock dataset for the purpose of the further development of security algorithms. Lastly, `v10` of Moriarty captures simulated device thefts. This is valuable information for developing continuous authentication solutions.

6. CONCLUSION

SherLock is the first publicly available long-term smartphone dataset designed for cybersecurity research. The dataset offers a wide variety of event-based and time series data, sampled at a rate that exceeds that of other publically available datasets. Moreover, collection of this data *did not require root privileges* or *changes to the operating system*, thus solutions based on this data are applicable to a large number of Android devices.

To demonstrate some of the dataset's usages, we performed a simple malware analysis using the explicit labels collected by Moriarty, and showed how the SherLock dataset can be used to evaluate continuous user authentication algorithms. We encourage the research community and security professionals to utilize this data for their research and development goals.

In the future, we plan to expand the experiment by adding more volunteers and versions of Moriarty. This data, in parallel with what is currently being collected, will open the door to additional research opportunities for the research community.

Acknowledgments. The authors would like to thank the scientific programmers: Simon Dzanashvili, Tomer Liokumovitch, Raz Shain, and Danny Shefer. This research was funded by the Israeli Ministry of Science, Technology and Space.

7. REFERENCES

[1] Trends (in) security everywhere. http://www.welivesecurity.com/wp-content/uploads/2016/01/eset-trends-2016-insecurity-everywhere.pdf, 2016. Accessed 5.16.

[2] It threat evolution q2 2015. https://cdn.securelist.com/files/2015/08/IT_threat_evolution_Q2_2015_ENG.pdf. Accessed 5.16.

[3] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android permissions: User attention, comprehension,

and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 3. ACM, 2012.

[4] S. Han V. Tendulkar B.G. Chun L.P. Cox J. Jung P. McDaniel A.N. Sheth W. Enck, P. Gilbert. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 2014.

[5] Shih-Hao Hung, Shuen-Wen Hsiao, Yu-Chi Teng, and Roger Chien. Real-time and intelligent private data protection for the android platform. *Pervasive and Mobile Computing*, 24:231 – 242, 2015. Special Issue on Secure Ubiquitous Computing.

[6] Rubin Xu, Hassen Saïdi, and Ross Anderson. Aurasium: Practical policy enforcement for android applications. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 539–552, 2012.

[7] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 2011.

[8] Gianluca Dini, Fabio Martinelli, Andrea Saracino, and Daniele Sgandurra. Madam: A multi-level anomaly detector for android malware. In *MMM-ACNS*. Springer, 2012.

[9] Liang Xie, Xinwen Zhang, Jean-Pierre Seifert, and Sencun Zhu. pbmds: a behavior-based malware detection system for cellphone devices. In *Proceedings of the third ACM conference on Wireless network security*, pages 37–48. ACM, 2010.

[10] Farrukh Shahzad, M Akbar, S Khan, and Muddassar Farooq. Tstructdroid: Realtime malware detection using in-execution dynamic analysis of kernel process control blocks on android. *National University of Computer & Emerging Sciences, Islamabad, Pakistan, Tech. Rep*, 2013.

[11] Takamasa Isohara, Keisuke Takemori, and Ayumu Kubota. Kernel-based behavior analysis for android malware detection. In *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*, 2011.

[12] Y. Pisetsky A. Shu D.S. Wallach M. Dietz, S. Shekhar. Quire: Lightweight provenance for smart phone operating systems. In *USENIX Security Symposium*, 2011.

[13] Guillermo Suarez-Tangil, Juan E Tapiador, Pedro Peris-Lopez, and Arturo Ribagorda. Evolution, detection and analysis of malware for smart devices. *Communications Surveys & Tutorials, IEEE*, 16(2):961–987, 2014.

[14] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. Andromaly: a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, 38(1):161–190, 2012.

[15] Hahnsang Kim, Joshua Smith, and Kang G Shin. Detecting energy-greedy anomalies and mobile malware variants. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 239–252. ACM, 2008.

[16] Asaf Shabtai, Uri Kanonov, and Yuval Elovici. Intrusion detection for mobile devices using the knowledge-based, temporal abstraction method. *Journal of Systems and Software*, 83(8):1524 – 1537, 2010. Performance Evaluation and Optimization of Ubiquitous Computing and Networked Systems.

[17] J. McClurg Y. Cao Y. Chen V. Rastogi, Z. Qu. Uranine: Real-time privacy leakage monitoring without system modification for android. In *Security and Privacy in Communication Networks*. Springer, 2015.

[18] W. Ng J. McClurg, J. Friedman. Android privacy leak detection via dynamic taint analysis. *EECS*, 2013.

[19] Jordan Frank, Shie Mannor, and Doina Precup. Activity and gait recognition with time-delay embeddings. In *AAAI*, 2010.

[20] Elaine Shi, Yuan Niu, Markus Jakobsson, and Richard Chow. Implicit authentication through learning user behavior. In *Information security*, pages 99–113. Springer, 2010.

[21] Jiang Zhu, Pang Wu, Xiao Wang, and Juyong Zhang. Sensec: Mobile security through passive sensing. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 1128–1133. IEEE, 2013.

[22] Nadav Aharony, Wei Pan, Cory Ip, Inas Khayal, and Alex Pentland. Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive Mob. Comput.*, 2011.

[23] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM conference on embedded networked sensor systems*, pages 71–84. ACM, 2010.

[24] Archan Misra and Lipyeow Lim. Optimizing sensor data acquisition for energy-efficient smartphone-based continuous event processing. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, 2011.

[25] O. Yurur and C.H. Liu. *Generic and Energy-Efficient Context-Aware Mobile Sensing*. CRC Press, 2015.

[26] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 153–162. ACM, 2006.

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[28] M. Bramer. *Principles of Data Mining*. Undergraduate Topics in Computer Science. Springer London, 2013.

[29] Android runningappprocessinfo. http://developer.android.com/reference/android/app/ActivityManager.RunningAppProcessInfo.html.

[30] Mobile security | lookout, inc. https://www.lookout.com/resources/reports/phone-theft-in-america. Accessed 5.16.

[31] Smart phone thefts rose to 3.1 million in 2013 - consumer reports. http://www.consumerreports.org/cro/news/2014/04/smart-phone-thefts-rose-to-3-1-million-last-year/index.htm. Accessed 5.16.

[32] A. Alzubaidi and J. Kalita. Authentication of smartphone users using behavioral biometrics. *IEEE Communications Surveys Tutorials*, 2016.

[33] Indrė Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010.

[34] Yisroel Mirsky, Bracha Shapira, Lior Rokach, and Yuval Elovici. pcstream: A stream clustering algorithm for dynamically detecting and managing temporal contexts. In *Advances in Knowledge Discovery and Data Mining*. Springer, 2015.

[35] M.H. Dunham, Yu Meng, and Jie Huang. Extensible markov model. In *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, pages 371–374, Nov 2004.

[36] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154, 2009.

[37] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[38] Daniel T. Wagner, Andrew Rice, and Alastair R. Beresford. Device analyzer: Large-scale mobile data collection. *SIGMETRICS Perform. Eval. Rev.*, 41(4):53–56, April 2014.

[39] Clayton Shepard, Ahmad Rahmati, Chad Tossell, Lin Zhong, and Phillip Kortum. Livelab: measuring wireless networks and smartphone users in the field. *ACM SIGMETRICS Performance Evaluation Review*, 38(3):15–20, 2011.

[40] Niko Kiukkonen, Jan Blom, Olivier Dousse, Daniel Gatica-Perez, and Juha Laurila. Towards rich mobile phone datasets: Lausanne data collection campaign. *Proc. ICPS, Berlin*, 2010.

[41] Anmol Madan, Manuel Cebrian, Sai Moturu, Katayoun Farrahi, et al. Sensing the" health state" of a community. *IEEE Pervasive Computing*, (4):36–45, 2012.

[42] Nathan Eagle and Alex Pentland. Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 2006.

[43] Tâm Huynh and Bernt Schiele. Analyzing features for activity recognition. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*. ACM, 2005.

APPENDIX

A. DETAILED DESCRIPTION OF THE SHERLOCK DATASET

Table 8: A description of the PUSH probes and their contents.

Probe	*Num. Fields*	*Description*
Call Log	5	Address, when, duration, outgoing or ingoing, and an indication if number is from user's contacts.
SMS Log	5	Address, when, outgoing or ingoing, and an indication if number is from user's contacts and if the content contains a URL.
Screen Status	2	Log of when the screen turns on or off.
User Presence	1	Android USER_PRESENT intent log: a record of when the user begins interacting with the device.
Broadcast Intents	3	All Android broadcast intents (events): changes in password, Bluetooth, network, RSSI, app packages, wallpaper, volume. Actions of button presses, picture/video taken, startup, shutdown, reboot, headset, phone ringing, notifications, TTS, and more.
App Packages	11	Log of when applications are installed, updated, or removed: provides the app's version, hash of the APK, and list of permissions.
Moriarty	6	All clues left by the Moriarty malware agent.

Table 9: A description of the PULL probes and their contents.

Probe	*Sample Interval*	*Sensors*	*Num. Fields*	*Description*
T0	1 day	Telephony Info	15	Information on the current telephony configuration.
		Hardware Info	6	The device's hardware configuration.
		System Info	5	Kernel, SDK, baseband, and general information.
T1	1 minute	Location	15	{longitude, latitude, altitude, (anonymized via clustering)}, speed, and accuracy.
		Cell Tower	5	Cell tower ID, type, and reception info.
		Device Status	14	Brightness, volume levels, orientation, and modes.
		WiFi Scan	4	**For each visible AP:** identifiers, encryption, frequency, and signal strength.
		Bluetooth Scan	9	**For each visible device:** identifiers, device class (type), parameters, and signal strength.
T2	15 seconds	Accelerometer	51	Statistics on 800 samples captured over a duration of 4 seconds at 200Hz. For each respective axis: mean, median, variance, covariance between axis, middle sample, FFT components and their statistics. A subset of these features is extracted from the orientation, rotation, and barometer sensors.
		Linear Accelerometer	51	
		Gyroscope	51	
		Orientation	9	
		Rotation Vector	12	
		Magnetic Field	51	
		Barometer	16	
T3	10 seconds	Audio	21	Statistics over 5 seconds.
		Light	3	Luminosity
T4	5 seconds	Global App Stats	98	Information on the CPUs, memory, network traffic, IO interrupts, and connected WiFi AP.
		Local App Stats	70	**For each running application:** statistics on CPU, memory and network traffic. Linux level process information from the system /proc folder.
		Battery	14	Configuration and statistics on power consumption and temperature.

Table 10: The data points collected by the *T4* probe every 5 seconds.

Local Application Statistics *(from each App)*		
Memory	Rss	
	rsslim	
	vsize	
	dalvikPrivateDirty	
	dalvikPss	
	dalvikSharedDirty	
	nativePrivateDirty	
	nativepss	
	nativeshareddirty	
	otherprivatedirty	
	otherpss	
	othershareddirty	
	Lru	
	minflt	
	cminflt	
	majflt	
	cmajflt	
CPU	cpu_usage	
	Utime	
	Stime	
	cstime	
	cutime	
	guest_time	
	cguest_time	
	num_threads	
	priority	
	starttime	
	Nice	
	itrealvalue	
	processor	
	rt_priority	
General	application name	
	package name	
	package uid	
	version code	
	version name	
Network	received bytes	
	received packets	
	sent bytes	
	sent packets	
Process	sid	
	pgid	
	pid	
	ppid	
	tpgid	
	flags	
	wchan	
	exit_signal	
	state	
	Importance *(Android)*	
	importanceReasonCode *(Android)*	
	importanceReasonPid *(Android)*	
	tcomm	
	startcode	
	endcode	

Battery			
	charge_type	level	status
	current_avg	online	technology
	health	plugged	temperature
	icon_size	present	voltage
	invalid_charger	scale	

Global Application Statistics			
Memory	MemFree *(Android)*	Active *(file)*	
	MemMax *(Android)*	Inactive *(file)*	
	MemTotal *(Android)*	Unevictable	
	MemUsed *(Android)*	Mlocked	
	MemTotal *(Linux)*	HighTotal	
	MemFree *(Linux)*	HighFree	
	Buffers	LowTotal	
	Cached	LowFree	
	SwapCached	SwapTotal	
	Active	SwapFree	
	Inactive	Dirty	
	Active *(anon)*	Writeback	
	Inactive *(anon)*	AnonPages	
		Mapped	
		Shmem	
		Slab	
		SReclaimable	
		SUnreclaim	
		KernelStack	
		PageTables	
		CommitLimit	
		Committed_AS	
		VmallocTotal	
		VmallocUsed	
		VmallocChunk	
Storage	Int. AvailableBlocks	Int. FreeBytes	Ext. BlockSize
	Int. BlockCount	Int. TotalBytes	Ext. AvailableBytes
	Int. FreeBlocks	Ext. AvailableBlocks	Ext. FreeBytes
	Int. BlockSize	Ext. BlockCount	Ext. TotalBytes
	Int. AvailableBytes	Ext. FreeBlocks	
CPU	current_cpuhertz		
	util. cpu 0,1,2,3		
	total cpu		
	tot_user		
	tot_nice		
	tot_system		
	tot_idle		
	tot_iowait		
	tot_irq		
	tot_softirq		
	ctxt		
	btime		
	processes		
	procs_running		
	procs_blocked		
Network	connectedWifi_ssid		
	connectedWifi_level		
	wifi rx bytes		
	wifi rx packets		
	wifi tx bytes		
	wifi rx packets		
	mobile rx bytes		
	mobile rx packets		
	mobile tx bytes		
	mobile tx packets		
	total rx bytes		
	total rx packets		
	total tx bytes		
	total tx packets		
IO Interrupts (counts)	msmgpio (*touch screen*)		
	wcd9xxx (*touch screen*)		
	synaptics_rmi4_i2c (*touch screen*)		
	cypress_touchkey (*back button*)		
	home_key		
	volume_down		
	volume_up		
	pn547	companion	
	SLIMBUS	sec_headset_detect	
	flip_cover	function_call_interrupts	