Towards Energy-Aware Intrusion Detection Systems on Mobile Devices

Monica Curti*, Alessio Merlo† *, Mauro Migliardi‡, Simone Schiappacasse∗

*Department of Informatics, Bioengineering, Robotics, and System Engineering (DIBRIS) - University of Genova
16145, Genova (Italy)
Email: moni890185@gmail.com, alessio.merlo@unige.it, megamone@gmail.com
† Department of Engineering - E-Campus University
22060, Novedrate (Italy)
Email: alessio.merlo@unicampus.it
‡Department of Information Engineering (DEI) - University of Padova
35131, Padova (Italy)
Email: mauro.migliardi@unipd.it

Abstract—This paper investigates the correlations between the energy consumption of Android devices and the presence of threats (e.g., battery-drain attacks). In particular, this paper proposes a model for the energy consumption of single hardware components of a mobile device during normal usage and under attack. The model has been implemented in a kernel module and used to build up an energetic signature of both legal and malicious behaviors of WiFi hardware component in different Android devices. Such activity allows us to build a tentative database of signatures that can be used to detect attacks by means of the actual energy consumption of a mobile device. The proposed power consumption model and kernel module can be applied also to other hardware components, so to obtain very precise energetic signatures.

I. INTRODUCTION

The importance of smartphones and tablet PCs is growing both in terms of market shares and in terms of possible uses. Although the size of this phenomenon may be related to a temporary fashion, it is possible to recognize that the growth of computing capability in small devices together with pervasive availability of broadband connectivity are promoting a transition of user interests toward a form factor that stresses availability without sacrificing mobility such as the one forecast in [1].

The proliferation of these devices has been seen by some researchers as the dawn of the era of a new generation of malware [5]; however, up to now several factors have limited this phenomenon. Among these factors we can cite the fragmentation of the OS landscape in the mobile world and the fact that mobile devices are mainly used as repositories of personal information with very limited and rather static functionalities, thus malware in a mobile device focuses on stealing agendas and address books while injecting parasitic behaviors has received a limited interest.

Recently, however, both these limiting factors have weakened significantly. First, the mobile landscape shows now a major growth of a single operating systems, namely Android OS (about 72% of the smartphones sold in 3Q12 according to a study by Gartner [6]); furthermore, the ability of Android-based smartphones to execute portable code in the Dalvik virtual machine [7] allows to easily overcome the high level of hardwares diversity. Second, the set of functionality available on a smartphone (or tablet PC) has grown immensely, thus this kind of devices is effectively substituting laptop PCs in several everyday usage patterns such as web-browsing and e-mail reading.

Furthermore, the last year has seen a significant penetration in govern agencies and public bodies. As a first example we can cite the recent security certification of Android smartphones by the US Department of Defense [8] that allows the deployment of Dell hardware with Froyo (Android OS v2.2) in the Pentagon. A second example is the adoption of tablet PCs (Apple iPad) by the Chicago hospital and the Loyola University Medical Center in Maywood. Finally, several research projects are focusing on the deployment of health-care services onto the tablet PC platform with widely goals from simple access to medical records [9], to reminders for medication intake [10], to decision support systems [11], to automatic recognition of pathological states [12].

The combination of these factors is radically changing the way security on mobile devices must be perceived. In fact, even if mobile malware taxonomies [4] are comprehensive, to date security on smartphones and tablet PCs is seen mainly as a privacy problem where personal data must be protected from theft and malicious exploitation. This fact is clearly shown both by the types of malware and by the types of check performed by mobile security applications to date.

We claim, on the contrary, that all types of threat that are usually considered for general purpose computing platforms must be considered for mobile platforms too. In fact, given the growing dependency of private users and public bodies from these kinds of devices, an attack aimed at disabling them (we will call it a Denial of Device DoD attack) is at least as dangerous as a data theft (see for instance [18]).

One of the more significant issues related to mobile devices is their complete dependency on battery power. In fact, if the battery is empty, the device is unusable. This weakness represents one of the facets of a new field of study called Green Security [13]. In fact both the specifics of battery drain attacks (a type of DoD attack) and the modeling of the device from its energy consumption point of view represent steps toward the definition of the capability of a device to manage...
In this work we cope with the detection of energy-based malware and attacks on Android. The corresponding state-of-the-art deals with two recent research trends, namely Android security and Energy-Aware Security. Regarding the first one, threats to the security of Android and to the end-user are being studied from architectural and applications perspectives. In the first case, Android Security mechanisms have been deeply analyzed (see [2] for some references) and improvements have been proposed (see [3]). In the latter case, some studies are devoted to the detection of malware hosted in Android applications and some solutions have been proposed but solutions in this research trend are currently limited to signature-based detection.

However, some OS-specific malware detection tools have been recently proposed. For instance, in [22] an Android specific malware detection tool is proposed, but, also in this case, the detection strategy is signature-based only and no discussion on energy-related attack is provided. In [20] and [19] solutions specific for Symbian and/or Windows Mobile OS are discussed. In particular, in [19] some early discussion on energy-related malware is provided.

Outside the mobile context, discussions on both signature-based and anomaly-based malware can be found in [21]; also in this case, there is no reference to energy-related malware. The field of Green, Energy Aware Security is younger than the Android security one, thus the available literature is far from being exhaustive in general. More in details, there are only very few works that focus on the identification of attacks relying on the energy consumption: in [14] a methodology for detecting three different kinds of attacks is provided; however the described methodology relies on external hardware measurement, without recording any measurement on the device itself. In fact, in order to avoid interfering in the mobility of devices, it is necessary to find solutions for acquiring consumption information directly from the device internals.

Such approach has been adopted by PowerTutor [15]; however, values provided by this tool, albeit sufficient to recognize some brute force battery-drain attacks [16], are not sufficiently precise nor reliable to build up signatures of energy-related attacks. In particular, the energy consumption calculated by PowerTutor is not the actual consumption but an estimation; in fact, it is not measured on the hardware, but it is coarsely assessed by means of average consumption value provided in XML files in the device configuration. Furthermore, it only takes into account the power consumption of data transmission, without considering the consumption related to the reception of packets.

The AppScope tool [17] is the only approach that seems to provide accurate energy consumption measurements. Unfortunately, such tool works only on a single model of mobile device (Google Nexus One) and it is not open-source, thus we were not able to replicate the approach on other models, nor evaluate the correctness of the experimental results. From a methodological point of view, AppScope adopts a mechanism for consumption analysis measurement that is strictly tied with the device model, by using static references. This makes hard to easily port the tool on other models.

Our approach overcomes such limitation since our measurement approach is based on drivers instead of device models, thus resulting in a more adaptable and scalable solution since the same driver (and battery) is used by different models (e.g. Sony Xperia U and Samsung Galaxy S Advance use the same battery drivers: ab8500).

III. A STEP-WISE APPROACH

The final aim of our proposal is the development of novel Energy-Aware Intrusion Detection solutions able to reliably detect attacks on mobile devices based on energetic footprints. To reach such result, we adopted a step-wise approach: firstly, we have developed some built-in solutions allowing to measure and analyze energy consumption directly on each device, neglecting the usage of external hardware; then, we performed measurements on some devices and benign/malicious applications and we started in populating a database of consumption patterns for both benign smartphone activities and known attacks.

The population of such database is a complex and in-progress task. The complete database is expected to become a repository of energy consumption signatures able to support the energy-based identification of malicious activities both directly (based on their specific energetic signatures) and indirectly (using anomaly detection techniques aimed at detecting variances on the average energy consumption of the device).

As a first step, we analyzed all the available energy consumption measurements inside an Android device. The value of the current energy consumption is provided by the battery module in each device. Such value represents the global instantaneous consumption and there is no way to retrieve the
contribution of each single device component from this global measurement. However, we were able to identify the hardware components whose consumption contribute to the global one provided by the battery module. Then, we modeled the global consumption as the sum of the consumption of each hardware component. Starting from this model, we focused on one specific component, namely the WiFi network module. Such choice is justified by a two-fold reason: firstly, literature shows that the WiFi component is the main target of battery-drain attacks (see [14], [15]). Secondly, we observed from official literature that such module (together with the 3G module) provides a significant contribution to the total energy consumption [23]. Then, we developed two different models, a global one - coping with all the hardware peripherals contributing in the energy consumption - and one specific for modeling the energy consumption of the WiFi module.

The general model formalizes the consumption in term of values of current and voltage. The voltage value follows a specific curve bounded with capacity that does not change according to the activity of the hardware components. Therefore, the only value that reflects the actual activities performed on the device is the electric current. Furthermore, we observed that the electric current value increases according to the discharge of the battery, in order to keep the power provided to all hardware components as much constant as possible.

The second model is specific for modeling the energy consumption of the WiFi module in terms of sending and receiving activities. However, such model is general enough that can also be applied to other network-related hardware components (e.g. the 3G module).

We have implemented this latter model in an ad-hoc kernel module that allows to calculate the power of both transmission and reception of the WiFi network module, without requiring the use of external hardware. By appraising energy consumption for the WiFi uplink and the downlink rates respectively, we were able to retrieve a value that expresses the consumption of a single byte both in transmission and reception. Therefore, this allow to measure the energy wasted in the communication at a single-byte precision.

We empirically evaluated our models as well as the measurement methodology, by using our kernel module to build the energy signatures of a normal device usage with benign testing applications and of a well-known network attack: the ping flood. Then, we repeated the attack at random times: our signatures allowed an easy detection of attack each time it has been performed.

We argue that by applying other models specific for other kinds of hardware components it is possible to get consumption values useful for the general model. Such model can be then used to create an energetic signature of both standard applications and attacks in terms of energy consumption of each main hardware components of the device. We are currently working on this extension which will provide more specific and precise energy signatures, related with an extended set of device hardware components.

At the end of this modeling and measurement phase, we plan to design and implement new Energy-based Intrusion Detection Systems able to recognize battery-drain and energy-based attacks on mobile devices, by means of both signature-based and anomaly detection strategies.

IV. Modeling Energy Consumption

As stated in the previous section, we propose here two models, i) a general consumption model, that takes into account energy consumption from all hardware components of a mobile device, and ii) a WiFi consumption model, specific for describing wireless network traffic both in terms of sending and receiving data.

A. General consumption model

We consider the global power consumption of a device as the sum of the power consumption of every single device component. In a smartphone, the main components are: CPU, Flash memory, Cellular radio, GPS, Graphics, LCD, Bluetooth, WiFi, Audio codec, Audio amplifier (see [23] for further details).

According to the Ohm’s Law, the power consumption is calculated as \( P = V \times I \). However, mobile batteries provide a variable voltage which decreases in a non-linear way (see Fig.1) as the battery discharges.

Mobile devices are made by components that require a minimum power level to work properly. Thus, since the value of the voltage decreases according to the discharge of the battery, in order to keep power as much constant as possible, the current level increases. In particular, we have observed that the voltage has a pattern which is inversely proportional to the value of the current.

This fact leads to the necessity to take into account the battery level during measurements. However, for the sake of simplicity, while we have taken it into account in the measurements, we do not refer to it explicitly in the following equations; on the contrary, we refer only to \( C_i \) as the power consumption of the i-th device component at a given battery capacity.

Given an activity on the device (e.g. file transmission) we model the global consumption as the sum of the contributions of all the hardware components related to the activity and
consumption which are independent from the same activity (we refer to it as "base consumption").

We define the global consumption \( C \) as:

\[
C = \sum f_i + g_i = B + P_s
\]

where:
- \( f_i = \) base consumption of the i-th component;
- \( g_i = \) consumption related to a specific (legal/malicious) activity for the i-th component;
- \( B = \sum f_i = \) base consumption of all components;
- \( P_s = \sum g_i = \) global consumption related to the specific activity.

By measuring the sum of the base consumption of all component (\( B \)) and subtracting such value from the global one measured in a given moment, it is possible to retrieve the consumption footprint of the specific activity (\( P_s \)). This activity may be a controlled legal or malicious one.

From a practical point of view, the global power measurement is complex and requires the development of proper software measuring modules for each hardware component. Such modules may require modeling the specific characteristics of the single hardware component.

A first step in this direction, we focus on the WiFi component, providing a specific model for measuring its consumption.

### B. Modeling the WiFi consumption

The general model allows isolating the power contributions of different hardware components that should be individually measured by ad-hoc modules. However, at present we can only measure the consumption of the WiFi component. Thus, to apply it to a real use case, we need to keep all the other components in negligible power consumption state. To overcome this limitation we implement another model that relates power consumption directly to transmission and reception of data. The construction of this model is based on a preliminary phase used to acquire information on power consumption related to the sending and receiving of data while keeping the consumption of the other components negligible.

To obtain these values, we execute two different kinds of tests, one for packet transmission and one for packet reception.

During tests, packets of the same size are continuously sent/received through the WiFi interface to measure the corresponding power consumption. Each round of test is related to a specific battery level (from 1% to 100% of the battery charge). Power values are instantaneous, and calculated as \( W = V \times I \); thus, we need to subtract the base consumption of all hardware components from the measured values to get the actual consumption related to packet transmission:

\[
P_{tx} = P_{tx \text{ test}} - B
\]

which corresponds to the power value in transmission calculated as transmission power during test (\( P_{tx \text{ test}} \)) minus the base consumption.

Then, we measure the number of packets actually transmitted by the kernel during the testing period, in order to retrieve the actual transmission time for a single packet as follows:

\[
T_{\text{tx pkt}} = \frac{1}{\text{packet rate}} = \frac{\text{measurement time}}{\text{transmitted packet}}
\]

Therefore, it is possible to obtain the energy value for the transmission (reception is symmetric) by multiplying such value for the measured power value and the number of packets:

\[
E(p) = E \times p = (P_{tx} \times T_{\text{tx pkt}}) \times p
\]

where:
- \( p = \) number of packets.
- \( E(p) = \) energy value for \( p \) packets.
- \( E = \) energy value for one packet.

Such preliminary phase allows retrieving standard values that can be reused in contexts where energy consumption related to other components is not negligible, as long as the packet size is the same used in the preliminary phase.

In this way, it is possible to experimentally calculate the energy consumption disregarding the energy consumption of other hardware components as long as the packet size is the same used in the preliminary phase.

### V. Measuring Battery Consumption

In order to provide a tool able to build up energy signatures of device activities, it is necessary to find reliable measures of the general power consumption.

#### A. Device selection

We first investigated which devices are able to provide both voltage (which is provided by almost every device) and electric current (which needs specific hardware). Actually, in order to have a suitable measurement inside the device, we need it to include a coulomb counter. We tested several devices to check the presence of this latter component: namely, Samsung Galaxy S, S3, Note2, HTC Desire HD and some less expensive like LG Optimus Chic, LG P350, Sony Xperia U and Samsung Galaxy S Advance; we found the coulomb counter only in two devices, namely the Samsung Galaxy S Advance and the Sony Xperia U. Furthermore, we discovered that it is necessary to use a specific battery driver (ab8500_fg) to obtain the value of the current.

Although every device has a battery driver that provides a current value, only those with the coulomb counter can provide a reliable one. In fact, those that do not have the coulomb counter simply obtain the current value by the relation expressed by the characteristic curve of the battery.

#### B. Driver modification

The kernel functions used to obtain the current measurements are not accessible from outside the battery driver; as an example, in Table I we show the visible fields in the Xperia U device. Thus, we had to export these methods, using the \texttt{EXPORT\_SYMBOL\_GPL(function\_name)} instruction inside the driver source code; this in turn requires a specific module that can execute these functions. The values are then transferred from kernel space to user space by means of a character device. More in details, we hooked through kprobes to the ab8500\_fg\_bat\_voltage to get a reference to
TABLE I. SONY XPERIA U VISIBLE FIELDS

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>POWER_SUPPLY_NAME</td>
</tr>
<tr>
<td>POWER_SUPPLY_VOLTAGE_NOW</td>
</tr>
<tr>
<td>POWER_SUPPLY_CURRENT_NOW</td>
</tr>
<tr>
<td>POWER_SUPPLY_CURRENT_AVG</td>
</tr>
<tr>
<td>POWER_SUPPLY_ENERGY_FULL_DESIGN</td>
</tr>
<tr>
<td>POWER_SUPPLY_ENERGY_FULL</td>
</tr>
<tr>
<td>POWER_SUPPLY_ENERGY_NOW</td>
</tr>
<tr>
<td>POWER_SUPPLY_CHARGE_FULL_DESIGN</td>
</tr>
<tr>
<td>POWER_SUPPLY_CHARGE_FULL</td>
</tr>
<tr>
<td>POWER_SUPPLY_CHARGE_NOW</td>
</tr>
<tr>
<td>POWER_SUPPLY_CAPACITY_LEVEL</td>
</tr>
</tbody>
</table>

the ab8500 struct and trigger ab8500_fg_instCurr_start and ab8500_fg_instCurr_finalize. The coulomb counter updates the value every 250ms, so we poll these functions at that rate to get a full profile of the energy consumption and use these values both in general model and in the first phase of the alternative WiFi model. In the latter case, we also need another module hooking two more kernel functions, namely netif_rx and dev_queue_xmit, to provide the instant transmission rate and the number of exchanged bytes.

VI. EXPERIMENTS AND RESULTS

In order to test out model we have built an experimental setup composed as such. We connect our device (namely the Sony Xperia U) through WiFi to a modem/router that supports a/g/n technology and, by means of an Ethernet USB On The Go adapter, to a second router. The Ethernet-USB adapter consumes a constant amount of energy so its contribution can be easily removed from the energy profile. Then we connect to the WiFi LAN a PC to be used to launch the attacks and to the wired LAN a second PC dedicated to the visualization of the measured values in a Matlab server. The separated LANs allow avoiding interaction between the attack traffic and at the same time have a real time visualization of the experimental data. In this way, we could refine our test routines and we automated its execution for the final measurements.

For the final measurements, only the WiFi connection was used. The measured values were written in log-files on the device and recovered at the end of the experiments.

This methodology allowed us to perform the experiments also on the Samsung Galaxy S Advance onto which “USB On The Go” technology is not available. We performed our tests several times. To minimize the Consumption due to activities different from the one we were actually measuring (see equations in section IV) we first run a script that kills all the related processes which we could not kill (e.g. inputmethod, systemui) without completely disabling the device. However, their energetic contribution is very low and it can be accounted for inside the Base Consumption.

The first phase of our experiments consisted in the construction of a database of energy signatures of applications and attacks.

More in details, on the application side we have generated the signatures for Skype, Shazam and YouTube, while on the attack side we have generated the signatures of 1) pingflood, that corresponds to illegal incoming data traffic, and 2) repeated HTTP GET requests made through kernel primitives, simulating a session of illegal outgoing data traffic. In order to check if the energy signature of contemporary activities is simply the sum of the energy signature of the single activities, we also generated the signatures for the combinations.

Once we got these signatures, we go through a monitoring phase in which our application continuously checks whether the current behavior of the application in foreground matches the specific one recorded in the database. In the case it does not, we check whether it matches a signature associated to a known attack.

We here report two specific use cases related to the general model: the first one is a Skype call subject to a ping flood attack (Fig. 2) while the second one is the play of a Youtube video and repeated HTTP GET requests (Fig. 3).

The first part of the figures shows both the raw measurements and the average of four values (a 1 second sliding window), the second part shows the average of 20 values (a sliding window of 5 seconds), and the third part shows the average of 40 values (a sliding window of 10 seconds). It is clear that both the raw measured values and the 1 second sliding window are too noisy to allow an easy identification of either behavioral changes in the energy profile of the system or the recognition of an energy signature. On the contrary, the 5 and 10 seconds sliding windows show a more stable behavior, definite trends and clearly identify the events that compose the experiment. In particular, in the following sections, describing two specific experiments, we adopt the values obtained using the 10 seconds sliding window.

A. Skype and pingflood

The first experimental example combines a Skype call with a ping flood attack.

More in detail(Fig. 2), we make a Skype call for 312 seconds then we trigger the attack. Immediately, we observe an increase of the current consumption from 652mA to 673mA. After 22 seconds the attack is consuming all the bandwidth and Skype loses the connection (the current goes down to 653mA); however, after a 17 seconds gap, the application tries to reconnect and the current jumps back up to the level measured at the beginning of the attack. After 252 seconds we close the Skype call and the current consumption, caused only by the pingflood attack, goes down to 620mA.

B. YouTube and HTTP GET

The second experimental example combines watching a YouTube video with repeated sending of HTTP GET requests (Fig. 3). More in detail, we first watch a video using the YouTube application for 290 seconds, then we trigger the attack. Immediately, we register an increase of the current from 690mA up to 780mA. During the attack, the measured current never stabilizes completely, however, it is always more than 730mA and most of the time above 760mA. Finally, after 300 seconds we close YouTube and the current goes down.

This experiment was quite long and it had a significant hit on the battery charge (battery level goes down from 40% to 15% of the maximum).
Our experiments clearly show that every activity has an immediate and noticeable impact on the energy consumption of the device. In fact, as the figures show, our measurements allowed us to easily identify the start and end time of the attacks as the energetic behavior of the device significantly moves away from the expected one.

Unfortunately, the measured values do not show a simple superposition of the effects, thus it is not possible to simply subtract the energy signature of the expected activity to have the energy signature of the activity unaccounted for (the attack). This lack of superposition may be due to the fact that, at present, we are not able to capture the power contribution of all the device components, thus a new evaluation will be performed once we have developed all the kernel modules needed.

Furthermore, even if no superposition exists at all, Android is not a general purpose OS. Thus, the possible combination of activities doesn’t suffer from combinatorial explosion and it would be necessary to identify the energy signatures only of a number of possible combination of foreground activity with background services.

In any case, our experiments clearly show that energy monitoring is a promising way for identifying unexpected (possibly malicious) behavior inside the Android operating system.

C. Discussion of the results

Our experiments clearly show that every activity has an immediate and noticeable impact on the energy consumption of the device. In fact, as the figures show, our measurements allowed us to easily identify the start and end time of the attacks as the energetic behavior of the device significantly moves away from the expected one.

Unfortunately, the measured values do not show a simple superposition of the effects, thus it is not possible to simply subtract the energy signature of the expected activity to have the energy signature of the activity unaccounted for (the attack). This lack of superposition may be due to the fact that, at present, we are not able to capture the power contribution of all the device components, thus a new evaluation will be performed once we have developed all the kernel modules needed.

Furthermore, even if no superposition exists at all, Android is not a general purpose OS. Thus, the possible combination of activities doesn’t suffer from combinatorial explosion and it would be necessary to identify the energy signatures only of a number of possible combination of foreground activity with background services.

In any case, our experiments clearly show that energy monitoring is a promising way for identifying unexpected (possibly malicious) behavior inside the Android operating system.
VII. CONCLUSION AND FUTURE WORKS

In this paper, we have defined a roadmap to energy-aware security and we have developed a specific use case to test our idea of energy-aware detection of attacks on mobile devices. In particular, we have discussed how to model energy consumption in Android devices in order to get the precise consumption of hardware components. We have applied our models to the calculation of the power and energy consumption of the WiFi hardware component. Furthermore, we have proved the efficacy of our models by calculating energetic signatures of a few applications and attacks and showing that power consumption allows identifying the attacks over the normal application behavior. Our work does not rely on external hardware for measurements thus does not impose any limitation on the device mobility.

At present, most of the research projects dedicated to the study of energy consumption on mobile devices are focused on power saving, while none (to the best of our knowledge) tries to relate energy consumption and security. Even if, at present, the limited number of effective malware for mobile devices may lead to the conclusion that there is no need for energy aware security, we argue that the technological and commercial trend toward a massive deployment of mobile, power constrained devices calls for a significant increase in the effort devoted to energy and security as a single field of study. In literature, it is possible to find a few works on energy aware security, but our approach can be considered new because we created models and tools allowing to generate and classify energetic signature both for regular and malicious activities, and, through this, detect potential attacks.

As future works, we aim to validate the models by means of more comprehensive experimental results and testing scenarios, as well as populating a wide database of signatures as input to the future development of Energy-aware Intrusion Detection Systems for mobile devices.

We are also aware that signature-based detection is not the only possible approach. In future developments we plan to study also an anomaly-based detection system.
REFERENCES


