2023 IEEE/ACM 10th International Conference on Mobile Software Engineering and Systems (MOBILESoft) | 979-8-3503-1182-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/MOBILSoft59058.2023.00018

On Security and Energy Efficiency in Android Smartphones

Joao Ferreira*¶‖, Bernardo Santos†‖, Wellington Oliveira‡, Nuno Antunes*, Bruno Cabral*, João Paulo Fernandes§
*CISUC & Dep. of Informatics Engineering, Universtity of Coimbra, Coimbra, Portugal
†LIACC &, Faculty of Engineering, University of Porto, Porto, Portugal
‡Dep. of Informatics, University of Lisbon, Lisbon, Portugal
§LIACC & Dep. of Informatics Engineering, Faculty of Engineering, University of Porto, Porto, Portugal
¶Colégio Agrícola Dom Agostinho Ikas da UFRPE, Federal Rural University of Pernambuco, Recife, Brazil
joao.ferreirasilva@ufrpe.br¶, up201706534@up.pt†, wdojunior@fc.ul.pt‡, {jfjunior, nmsa, bcabral}@dei.uc.pt*, jpaulo@fe.up.pt§

***Abstract*— Smartphones are so immersed in our everyday lives that it is hard to imagine our routines without them. For most of us, they have already replaced alarm clocks, calculators, organizers, maps, and countless other things. As such, running out of battery charge means losing all of these functionalities, not just the inability to communicate using calls or messages, which can already be critical. The widespread usage of mobile devices is accompanied by a growing number of cybercriminals exploring scams for (illicit) benefits. Indeed, the more data flowing through mobile devices and apps, the greater the possibility of exploring threats and attacks. Users are then concerned about unintended access to critical data such as sensitive stored information, bank accounts, passwords, social media accounts, or private files. Although documented strategies exist to mitigate security risks, implementing the corresponding security mechanisms may impose an overhead on energy consumption, which in practice, affects the device's battery charge. In this paper, we analyse the impact of security mechanisms on energy consumption in the context of Android mobile devices. We investigate the energy consumption of operations such as copying files and logging in with and without encrypting the credentials. Our results quantify the energy overhead of certain security mechanisms and confirm that there is a statistically significant increase in energy consumption when security standards, e.g., data encryption, are adopted. This work highlights the need for understanding the trade-offs between energy consumption and security in mobile devices and serves as a reference for mobile application developers to consider energy efficiency when implementing security measures.**

***Index Terms*—Energy Consumption, Mobile Security, Mobile Device, Android**

I. Introduction

Mobile phones have become an essential part of modern life and are present in almost every aspect of our daily lives. In the last decade, they have evolved rapidly regarding technology and capabilities. These devices can perform various and diverse tasks, in addition to simply making and receiving calls, including communication, navigation, payments, entertainment, and internet access [1].

Most of the world's population has access to mobile devices. The total number of people who own a smartphone is 7.33 billion [2]. The adoption rate of smartphones worldwide is over 100%, meaning there are more smartphones in use than people on the planet. Furthermore, it is estimated that people actively use smartphones for an average of 5 hours a day. In this context, the battery life of these mobile devices is an increasingly important factor concerning consumer satisfaction [3].

Due to the widespread usage of mobile devices worldwide, especially smartphones, the interest of criminals in invading these systems to commit illicit actions is also increasing. Since mobile apps process personal data, access sensitive information, and sometimes allow remote control of devices (like IoT equipment), they become desirable targets. Smartphones, widely used as IoT devices, play a fundamental and crucial role in the business models of many innovative companies, including those operating in the FinTech and HealthTech sectors, among others [4].

Smartphones can be vulnerable to various security threats, including sniffing, spam, attacker spoofing, phishing, pharming, vishing, and data leakage [5]. Strategies exist that can protect the device and the data in it, mitigating these threats, such as the use of encryption; some strategies target the end-user, some target the application developer, and others are server-side.

The Open Web Application Security Project[1] (OWASP) is an international organization that focuses on improving software security. This organization provides resources and tools to help developers, security professionals, and organizations create and maintain secure applications. But, the impact of these tools and techniques on energy consumption is yet to be characterized.

We aim to investigate the effects of utilizing encryption on powering mobile devices. In particular, we seek to address the following questions:

- **RQ1:** Does implementing encryption have an impact on the energy usage of smartphones?
- **RQ2:** How does energy consumption change when executing encryption-related tasks, such as file copying and logging in?

To measure the impact of the security techniques suggested by OWASP (cryptography), three scenarios were implemented

‖The first two authors contributed equally to this work

[1]https://owasp.org/. Accessed 12th February 2023.

979-8-3503-1182-2/23/$31.00 ©2023 IEEE
DOI 10.1109/MOBILSoft59058.2023.00018

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on February 25,2024 at 19:14:40 UTC from IEEE Xplore. Restrictions apply.

and assessed using a tailor-made Android application.

As a result of incorporating an additional layer of security in the form of cryptography, our hypothesis would be to assume an increase in energy consumption. In addition to this natural assumption, it is important to stress that we were able to quantify this increase. We consider cases where it will be necessary to perform a high volume of operations, where a difference in energy consumption can be quite considerable.

An experiment was conducted to compare the energy consumption using cryptographic methods in Android application development. As mentioned before, we developed experiments for three scenarios: logging in using credentials stored in a common file, logging in using the SharedPreferences API, and copying a common file. These scenarios cover four security risks (M1, M2, M4, and M5, presented in detail in section II-B), and for each scenario, we implemented an encrypted and unencrypted version of the code.

The measurements collected show statistically significant differences in energy consumption. As expected, the encrypted version consumes significantly more than its unencrypted counterpart, the most extreme being an eleven times increase when performing 50,000 logins using SharedPreferences.

The remainder of the paper is organized as follows: Section II introduces background concepts. Section III describes the methodology we adopted. Section IV presents the experiments and results. Section V presents the related works, and finally, Section VII concludes the paper and suggests directions for future work.

II. Background

This section discusses background concepts needed to contextualize our work and motivate some of our design decisions. In particular, we present an overview of the mobile environment (Section II-A), mobile security (Section II-B), cryptography (Section II-C), energy consumption in mobile devices (Section II-D), and tools to measure this consumption (Section II-E).

A. Mobile Environment

A mobile development environment is a set of tools and platforms to develop applications for mobile devices such as smartphones and tablets. It includes development software, SDKs (software development toolset), libraries, and other resources that help developers create and test applications for mobile operating systems such as iOS or Android. A mobile development environment can include tools for code development, project management, systems integration, automated testing, and device emulation, among others [6].

Our study focuses on Android, an open-source operating system developed by Google based on a modified Linux kernel. This design decision was driven by the fact that Android is currently the world's most widely used mobile operating system, with a market share of over 70% [7].

Android provides a framework for developing mobile applications using Java or Kotlin, with built-in support for numerous standard features, such as GPS and Bluetooth, and security features, like authentication or access control. Like any other complex software, it also has its own set of vulnerabilities (insecure data storage, server-side vulnerabilities, injection flaws, or insecure direct object reference attacks), and the misuse of its built-in functionalities may result in vulnerable applications [4].

B. Mobile Security

The security of computing devices has always been debated in academic and industrial communities. With the increase in the use of mobile devices, research in the niche of mobile device security has been highlighted [8]–[10].

Application security initially varies depending on the device's operating system (iOS or Android). The security server APIs are generally more difficult to invade than the front-end, but this depends on the technical and functional context of the application itself [4]. In the business industry, it is notorious that many software security companies also provide protection solutions for mobile devices. The market leaders are Symantec[2], McAfee [3], Kaspersky[4], Norton[5], Trend Micro[6], and Lookout[7].

The OWASP (introduced in Section I) is a foundation that works to improve the quality of security software. Over the years, they have published software security risk lists in various areas, such as privacy, client-side, low-code platforms, or cloud computing. One published list identifies the ten biggest security risks of mobile devices [11], described in Table I. The OWASP list was chosen because it has no financial relationship with companies that sell software with security services. Its projects, tools, documents, forums, and chapters are free and open to anyone interested in improving application security. Furthermore, provide us with clear examples of possible attack scenarios and how to protect against these risks.

TABLE I
OWASP TOP TEN MOBILE VULNERABILITIES (2016)

M01	Improper Platform Usage
M02	Insecure Data Storage
M03	Insecure Communication
M04	Insecure Authentication
M05	Insufficient Cryptography
M06	Insecure Authorization
M07	Client Code Quality
M08	Code Tampering
M09	Reverse Engineering
M10	Extraneous Functionality

To better manage risks, we categorize them based on their area of impact. Development risks are grouped under M01,

[2]https://www.broadcom.com. Accessed 2nd March 2023.
[3]https://www.mcafee.com. Accessed 2nd March 2023.
[4]https://www.kaspersky.com. Accessed 3rd March 2023.
[5]https://www.norton.com. Accessed 4th March 2023.
[6]https://www.trendmicro.com. Accessed 4th March 2023.
[7]https://www.lookout.com. Accessed 4th March 2023.

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on February 25,2024 at 19:14:40 UTC from IEEE Xplore. Restrictions apply.

M02, M04, and M05, server-related risks under M03 and M06, and software engineering risks under M07, M08, M09, and M10.

C. Cryptography

Cryptography is the practice of secure communication and data storage. It involves using mathematical algorithms to encrypt and decrypt information, making it unreadable to unauthorized parties. Thus, it protects sensitive information, such as user login credentials or other personal information. There are two main types of cryptography: symmetric and asymmetric-key. In the first, the same key is used for encryption and decryption, while in the latter, two different keys are used, a public for encryption and a private for decryption.

In this work, we leverage three different cryptographic algorithms to fix a set of vulnerabilities: AES, CBC, and PKCS5PADDING. Advanced Encryption Standard (AES) [12] is a widely used symmetric-key algorithm to encrypt and decrypt data. Cipher Block Chaining (CBC) [13] helps ensure that identical plaintext blocks are encrypted to different ciphertext blocks by combining an initialization vector with the first block of plaintext before encryption. Furthermore, PKCS5PADDING [14] is a padding scheme that ensures the encrypted blocks are the correct length by adding additional bytes to the plaintext before encryption.

D. Energy Consumption in Mobile Devices

In mobile devices, energy consumption is a critical concern due to the limited power resources of these devices. Although there is no standard for smartphone use, on average, some pieces of hardware dominate the power consumption in a smartphone, such as the processor, networking, and screen [15]. However, the software also plays a crucial role in determining the energy consumption of a mobile device, as it is responsible for instructing the hardware on what tasks to perform [7].

It is important to increase battery life on mobile devices, as 90% of users say they suffer from anxiety when their devices reach low battery levels, having to keep up with the deactivation of some features [16]. The fear of being incommunicado, related to anxiety about running out of smartphone battery, is on the verge of becoming a clinical condition called nomophobia [17].

How software is designed and implemented can significantly impact a mobile device's energy consumption. Previous works have analysed the impact of some of these design decisions, such as programming languages [18], machine learning techniques [19], and UI elements [20]. For example, poorly designed software can result in the CPU being active even when the device is idle, resulting in unnecessary energy consumption. Similarly, a poorly designed communication protocol can result in excessive energy consumption during data transfer [21].

E. Power consumption analysis tools

During the development of this work, several tools were studied that analyse the energy consumption of mobile devices; we highlight AnaDroid, Batterystats, E-MANAFA, and Ebserver.

- **AnaDroid** is a tool to monitor and analyse the energy consumption of Android apps. AnaDroid was developed in the context of the Green Software Lab project [22].
- **Batterystats** is a tool that belongs to the Android framework, capable of collecting data from devices. It is common to use ADB to create a report that can be analysed using Battery Historian [23].
- **E-MANAFA** energy profiler is a plug-and-play, device-independent, model-based profiler able to acquire fine-grained energy measurements on Android devices [24].
- **Ebserver** is a tool to automatically collect the power consumption of Android devices through the ADB BATTERYSTATS [25].

III. Methodology

In this section, we present the methodology employed in our research to assess the overhead of security mechanisms on the energy consumption of Android smartphones. Figure 1 depicts an overview of the applied methodology.

For the purposes of our work and our line of research, we need an automatable mechanism that allows us to compare mobile applications using specific security resources and analyse the respective energy consumption (Figure 1 - System Understanding). We developed an application[8] (Figure 1 - Mobile Application), for which the requirements were established to ensure precise control of all operations performed on the device and differentiate the origin of variations in energy consumption. The application was developed to allow a user-defined number of repetitions for an operation, such as user authentication through login and file copying, whether performed with or without the implementation of encryption mechanisms. This application was developed using the Java programming language and the Android Studio development environment.

Regarding the analysis of energy consumption, we identified Ebserver as an adequate tool for our needs (Figure 1 - Ebserver). The three criteria were used to guide our choice:

(i) small amount of instrumentation;
(ii) automatic execution;
(iii) capability to pinpoint the energy consumption of each process.

Ebserver presents these three characteristics and has been previously used to analyse the Android device's energy consumption in a different context, such as development approaches [26], Java collections [27], complex event processing and reactive languages [28] and multi-language offloading [29]. In addition, the Ebserver leverages HTTP requests between the application and a node.js server application to execute the following steps:

(i) send to the application information about the app or experimental scenarios that should be executed;

[8]https://github.com/greenstamp-project/seeas-app/. Accessed 16th March 2023.

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on February 25,2024 at 19:14:40 UTC from IEEE Xplore. Restrictions apply.

(ii) log the resource usage from the app execution (i.e., energy, CPU, and memory usage);
(iii) close the application.

Furthermore, it is possible to measure the device's consumption after several executions in different operating conditions and natively integrate with Android applications developed in Java. The results are recorded in structured text files, facilitating their subsequent analysis and interpretation.

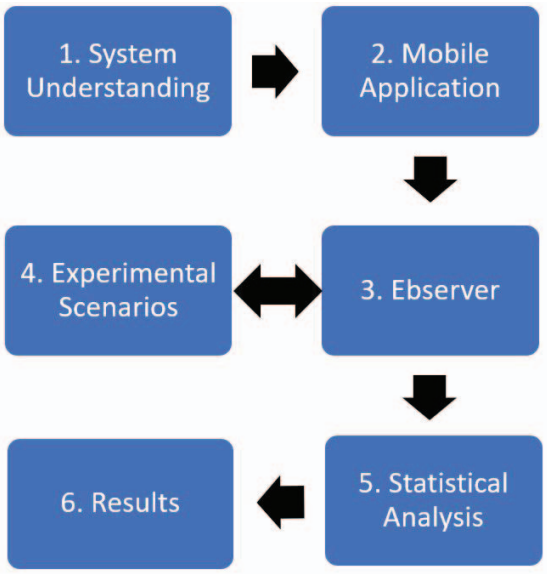


Fig. 1. Methodology adopted

To assess the impact of security mechanisms on the power consumption of Android smartphones, we designed a variety of experimental scenarios covering the main threats to smartphone security, including improper platform usage, insecure data storage, insecure authentication, and insufficient cryptography (Figure 1 - Experimental Scenarios) (Presented in detail in section IV-A).

In the subsequent phase of statistical analysis (Figure 1 Statistical Analysis), we investigated the existence of significant relationships between the values obtained by verifying the independence of the values, the normal distribution of the data and the homogeneity of the variance. These checks are essential to ensure the validity of the analysis of variance (ANOVA) to be performed later (Further details in section IV-B).

Finally, we developed Python scripts to collect and analyse the relevant data (Figure 1 - Results), including energy consumption and the execution time of experimental scenarios, from the files generated by Ebserver. It performs a series of statistical tests, calculates metrics, and generates graphs that provide a complete analysis and allow comparison between the results obtained. The tool and all data collected are available [9].

[9]https://github.com/greenstamp-project/seeas-stats/tree/1.1. Accessed 16th March 2023.

In the next section, we present an experiment, with the steps described at a high level in this methodology. The experiments were conducted on a Redmi Note 10 Pro Max[10], featuring a Kryo 470 Octa Core @ 2.3GHz processor and 6 GB of LPDDR4x RAM, running on Android 11. The experimental scenarios were developed using Android Studio version 201.6719854 and Android API 16.

IV. Experiments and Results

In this section, we present the experiments and results of this work. Aiming at a better organization, we subdivided it into experimental scenarios, statistical tests, and results.

In our experiments, we aim to measure the impact that using security mechanisms has on the energy consumption of mobile devices. At first, it was necessary to identify the biggest security risks on mobile devices. We follow the list of the ten biggest security risks published by OWASP [11] subject to mobile devices.

The goal of this work was to select practical functions that could be reproduced on the developer's side. Therefore, our study could reproduce at least one scenario representing a security risk of categories M01, M02, M04, and M05, proposed by OWASP (Detailed in Section II-B). The other categories are beyond the scope of this work, either because they are classified as software engineering (M07, M08, M09, and M10) or are more focused on security issues implemented on the server side (M03 and M06).

A. Experimental Scenarios

As mentioned in Section III, three categories of experimental scenarios were developed to verify if there is any relationship between security issues on android devices and their energy consumption. For each category, we performed tests with varying amounts of runs and file sizes, presented in Table II.

TABLE II
Experimental scenarios categories, repetitions, and file sizes. N/A stands for not applicable

Category	Repetitions	File Sizes (MB)
Local File Login	100	N/A
	1000	N/A
	5000	N/A
	10000	N/A
SharedPreferences Login	1000	N/A
	10000	N/A
	50000	N/A
Store Files Locally	10	10, 20, 30
	30	10, 20, 30
	50	10, 20, 30

Each operation was performed 30 times, and the obtained average values were considered. For example, we made ten copies of a file size of 20 MB to generate one result. We repeated this operation 30 times and considered the mean. Following the same principle, we do 50,000 logins using SharedPreferences credentials and get one result.

[10]https://www.mi.com/in/redmi-note-10-pro-max/specs. Accessed 13th March 2023.

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on February 25,2024 at 19:14:40 UTC from IEEE Xplore. Restrictions apply.

B. Statistical Tests

An adequate statistical analysis of the data is necessary to reach meaningful conclusions; this section describes the statistical tests performed on the collected data in detail.

First, the outliers were removed using the interquartile method. Then, to ensure that the samples collected from each version of the experimental scenarios were comparable and that the difference between the version with cryptography and without them was not due to chance, we performed an analysis of variance (ANOVA) test [30]. Three assumptions must be checked before performing this test:

1) samples are drawn independently
2) data is normally distributed
3) homogeneity of variance

Our methodology ensures that the first assumption is respected. To check for normality, we perform three normality tests per experimental scenarios version: Shapiro-Wilks [31], D'Agostino's K^2 [32] and Anderson-Darling [33]. Only experimental scenarios that pass all three tests are normally distributed and therefore tested for homogeneity of variance using Levene's test [34].

Experimental scenarios that do not respect all assumptions of ANOVA are tested using random sampling [35], or A/B testing, with 500,000 repetitions. This test has no assumptions and indicates whether or not a difference in the groups' means is due to chance. The sample differences were statistically significant for ρ-value ≤ 0.05.

C. Results

Here, we present the results of our experiments divided into the respective experimental scenarios categories. All the subsections follow the same structure: plot the data without outliers, determine whether the differences in the measurements are statistically significant, and calculate the mean percentage differences.

1) ***Local File Login****:* In this first scenario, N login operations were executed by reading the access credentials from an ordinary file in the Android registers (Table III). The access credentials used were: a common e-mail address and a strong password, with special characters and uppercase and lowercase letters. In all experiments, the user was "student@dei.uc.pt", and the password was "GreenStamp2@23".

In Figure 2, we present the plotted measurements from all the experimental scenarios in this category. With a quick analysis of the plots, we verify that the energy consumption of the non-vulnerable, i.e., encrypted, version of the experimental scenarios is larger in every case, independently of the number of iterations performed.

We calculated the average of the measurements for both versions of each experimental scenario and the percentage increase between the vulnerable and non-vulnerable versions, presented in Table III. As seen in Figure 2, independent of the number of iterations, the non-vulnerable version of the experimental scenarios always consumed more energy, and the statistical tests indicate that these consumption differences are not due to chance. It is also possible to notice that an increase in iterations results in a greater percentage change in energy consumption.

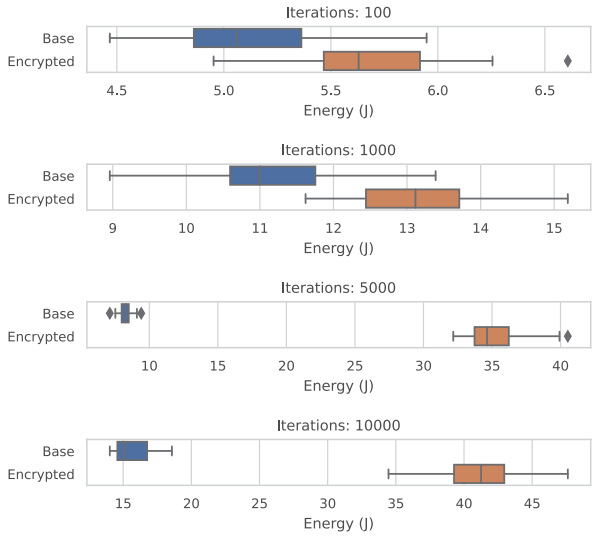


Fig. 2. Energy consumption of the base implementation of the local login experimental scenarios and the encrypted version, with 100, 1,000, 5,000, and 10,000 iterations.

TABLE III
MEAN ENERGY CONSUMPTION OF ALL LOCAL LOGIN EXPERIMENTAL SCENARIOS VERSIONS, PERCENTAGE INCREASE BETWEEN THE BASE AND ENCRYPTED VERSIONS, AND STATISTICAL SIGNIFICANCE OF THE DIFFERENCES

Iterations	Base (J)	Encrypted (J)	Sig. Diff.	% Increase
100	5.13	5.69	✓ (ANOVA)	10.88
1,000	11.15	13.15	✓ (ANOVA)	17.99
5,000	8.28	35.10	✓ (A/B)	323.96
10,000	15.63	41.30	✓ (A/B)	164.26

By increasing the number of operations, energy consumption was expected to increase. According to Table III, this is true for the encrypted versions of the experimental scenarios. However, when executing the base version 5,000 times, the energy consumption was significantly inferior to when executing it 1,000 times. Consequently, the increase in consumption for performing 5,000 logins was 323.96%, much larger when compared to the other tests.

2) ***SharedPreferences Login****:* In this second scenario, N logins were performed by reading the access credentials from a file in the Android registers (presented in Table IV). Unlike the previous scenario, this file is native to Android and specific for this operation. Its execution is faster compared to reading a common file. For this reason, it was possible to carry out more iterations. The same credentials were used.

The measurements collected for the various experiments using the SharedPreferences experimental scenarios are presented in Figure 3, where we can verify that the version of the application encrypting data before storing always consumes

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on February 25,2024 at 19:14:40 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
MEAN ENERGY CONSUMPTION OF ALL SHAREDPREFERENCES LOGIN EXPERIMENTAL SCENARIO VERSIONS, PERCENTAGE INCREASE BETWEEN BASE AND ENCRYPTED VERSIONS, AND STATISTICAL SIGNIFICANCE OF THE DIFFERENCES

Iterations	Base (J)	Encrypted (J)	Sig. Diff.	% Increase
1,000	2.10	2.72	✓ (ANOVA)	29.93
10,000	2.36	9.17	✓ (ANOVA)	288.99
50,000	2.28	28.23	✓ (A/B)	1137.81

more energy.

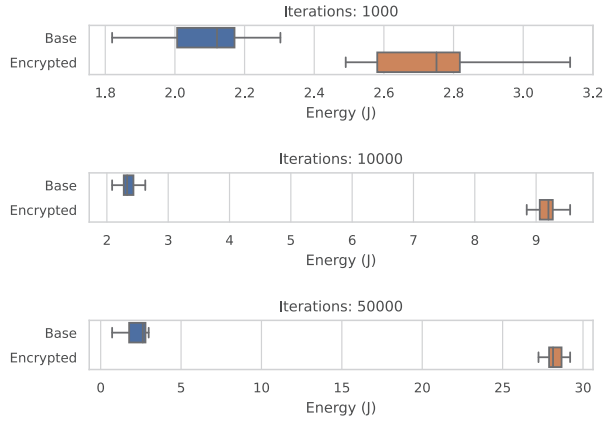


Fig. 3. Energy consumption of the base implementation of the SharedPreferences login experimental scenarios and the encrypted version, with 1,000, 10,000, and 50,000 iterations.

Table IV presents the mean energy consumption of each experimental scenario, the percentage increase, and indicates whether or not the difference is statistically significant. Analysing this data, we see that the consumption of the non-encrypted version does not vary much with the number of iterations, but the encrypted version's consumption, on the other hand, suffers a large increase. The differences in consumption between the base and encrypted versions show a statistically significant difference and grow larger as the number of iterations increase, reaching a 1137.81% increase when executed 50,000 times.

*3) **Store Files Locally**:* In this third scenario, we look at copying text files with three different sizes, 10 MB, 20 MB, and 30 MB. The files were stored in the Downloads directory, where it is possible to make many copies of files without major problems due to Android security restrictions.

Starting with an analysis of the plotted data, presented in Figure 4, we can verify that the encryption versions always consume more energy than the base versions. Furthermore, the differences in consumption appear to grow larger as the number of iterations increases.

The results of the statistical tests, presented in Table V, indicate that the encrypted versions of the benchmarks do consume more energy than the base versions and that these consumption differences are statistically significant. We also verify that, for the same file, increasing the number of iterations translates to an increase in the energy consumption difference. Moreover, we can conclude that storing a larger file consumes more energy for the same number of iterations.

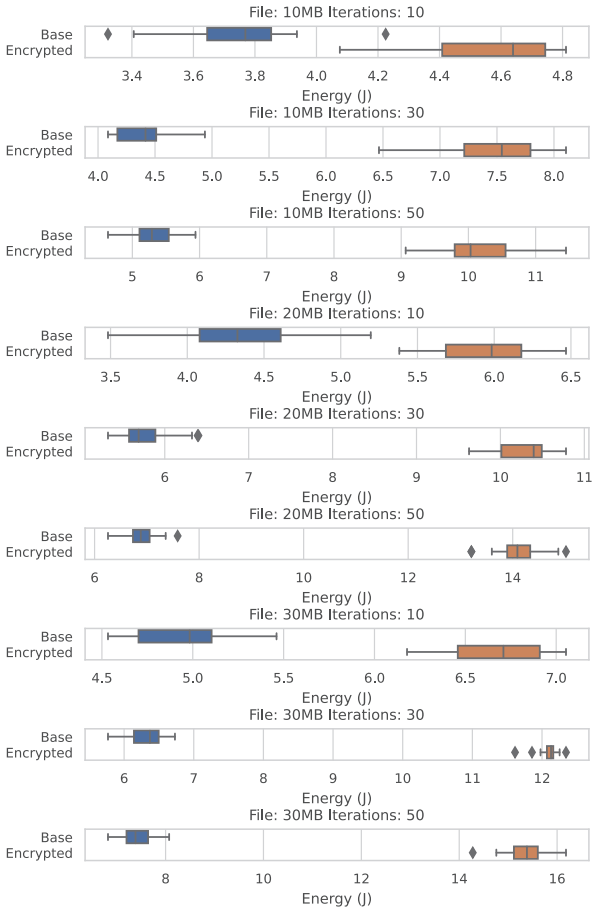


Fig. 4. Energy consumption of the base implementation of the local file storage experimental scenario and the encrypted version, with different file sizes and number of iterations

TABLE V
MEAN ENERGY CONSUMPTION OF ALL LOCAL FILE STORAGE EXPERIMENTAL SCENARIOS VERSIONS, PERCENTAGE INCREASE BETWEEN BASE AND ENCRYPTED VERSIONS, AND STATISTICAL SIGNIFICANCE OF THE DIFFERENCES

File Size (MB)	Iter.	Base (J)	Encrypted (J)	Sig. Diff.	% Inc.
10	10	3.74	4.56	✓ (ANOVA)	21.81
	30	4.41	7.44	✓ (A/B)	68.62
	50	5.30	10.20	✓ (ANOVA)	92.40
20	10	4.32	5.96	✓ (ANOVA)	38.10
	30	5.75	10.28	✓ (A/B)	78.90
	50	6.91	14.11	✓ (ANOVA)	104.25
30	10	4.96	6.66	✓ (ANOVA)	34.48
	30	6.35	12.08	✓ (A/B)	90.38
	50	7.42	15.33	✓ (ANOVA)	106.50

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on February 25,2024 at 19:14:40 UTC from IEEE Xplore. Restrictions apply.

V. Related Works

In recent years, research has been done to analyse the energy consumption of several areas of computing. In this section, we emphasise some recent works related to the energy consumption of mobile devices or mobile security.

It is important to mention Mazuera-Rozo et al. [36] work, which investigated the 681 commits, including security weaknesses fixed by developers in Java and Kotlin apps, to define a taxonomy highlighting the types of software security weaknesses. In addition, they realized a survey with Android developers to validate the proposed taxonomy. The proposed taxonomy features 80 software security weaknesses from a mining-based study. It is important to stress that their article inspired us to develop this research, looking at energy consumption associated with security issues of mobile devices.

Tawalbeh et al. [37] presented an analysis of the energy consumption of several components of a device, namely 3G, Wi-Fi, OLED, and GPS, among others. Devices from different manufacturers were compared using two measurement tools, PowerTutor and AmobiSense. They concluded that the Galaxy Note 3 device consumes more power than the Sony Xperia Z2. Furthermore, they claim that the Wi-Fi and 3G units consumed about half the total energy consumed by all other components.

Hoque et al [6] presented several energy consumption profiles for mobile devices. In addition, they introduce terminology and describe energy modelling and measurement procedures applied in model-based energy profilers. Then, rate profilers on implementation and deployment strategies and compare profiling capabilities and performance across multiple types. In the end, they address the limitation and challenges of each one.

Gregório et al. [38] presented a study where they looked for known energy consumption patterns in applications. For this, they decompiled applications and analysed the resulting code. They investigated 236 open-source applications. Next, it compared the presence of energy patterns between source code and decompiled code. Finally, conclude the work by pointing out that patterns were detected in 69% of the analysed applications, both in the source code and in the APK.

You et al. [39] analyses the difference in energy consumption of mobile devices when transferring the processing of some operations to clouds located at the edges of cellular networks. The allocation of resources for a multiuser MECO system based on time division multiple access (TDMA) and orthogonal frequency division multiple access (OFDMA) was considered. They conclude that the optimal resource allocation policy must have a limit-based structure to minimize the weighted sum of mobile energy consumption.

Wang et al. [5] introduced four testing approaches to mobile security: mobile forensics, penetration testing, static analysis, and dynamic analysis. In addition, they set up their network, carrying out mobile security tests in the following axes: identifying confidential data on a mobile device, performing penetration tests on mobile devices, and detecting data leakage.

Weichbroth et al. [40] conducted a literature review to identify and analyse existing threats and best practices for security on mobile devices. They found recognizable threats, established best practices, and surveyed 167 mobile app users. Most respondents stated that using dual methods reduces the security risk. The authors warn that new architectures are needed, as we are close to a new era of cybercrime, with the use of 5G networks, artificial intelligence, and biometrics.

The study by Yao et al. [41] was developed through Kano quality attributes, mobile security application resources, and their influence on user satisfaction. The authors verify the main security features with the greatest impact on user satisfaction. They traced relationships between the security issues that most please men and women, and between users with high and low familiarity with technology.

Montenegro et al. [42] guide Android application developers regarding the power consumption of cryptographic primitives and help them choose the most energy-efficient cryptographic algorithms. The study focuses on three providers (IAIK, BC and SC). The research questions of this work are directed to server-side encryption issues and not to the development side, the focus of our work.

Merlo et al. [43] explore the possibility of using power-consumption-based sensors to detect security threats, such as battery-drain attacks, on mobile devices running on the Android platform. The paper suggests a measurement methodology and high-level models for assessing the energy consumption of two crucial hardware subsystems: the CPU and Wi-Fi. To demonstrate the effectiveness of the proposed methodology, the researchers conduct an energy-oriented version of a ping-flood attack while a legitimate application runs. Unlike our work, this one proposes a methodology for measuring energy consumption and considers CPU and Wi-Fi.

TABLE VI
Scope of related work

Authors	Energy Consumption	Security
Mazuera-Rozo [36]		✓
Tawalbeh [37]	✓	
Yao [41]		✓
Gregrório [38]	✓	
Weichbroth [40]		✓
You [39]	✓	
Wang [5]		✓
Hoque [6]	✓	
Montenegro [42]	✓	✓
Merlo [43]	✓	✓
Our Proposal	✓	✓

Table VI presents a summary of the existing duality in searches related to mobile devices. One group focused on security issues, and the other focused on energy consumption issues. We have identified two works that cover both areas

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on February 25,2024 at 19:14:40 UTC from IEEE Xplore. Restrictions apply.

of interest. However, upon closer examination, we determined that their scope differs from ours.

VI. Threats to Validity

In this section, we address potential threats to the validity of the research. Identifying and evaluating these factors is crucial for ensuring the reliability and validity of the results.

The outcomes introduced in this paper are from the execution on a single device. Past works have looked into energy utilization of various multiplatform application development [44] or programming language [26], [45] utilizing numerous mobile devices indicate that similar relations are expected, independently of the number of devices used. Regardless, our outcomes might change, assuming the experiments presented here were to be executed on different devices.

Although Kotlin is the Android official programming language [46], the application used throughout this work was written in Java. We opted to analyse the impact of Java safe constructs because over 99% of the Android applications are still written in Java [45]. No measure was conducted to analyse the energy impact of Kotlin safe methods.

Ebserver was used to collect the energy measurements presented here. This tool has been used before to collect energy data from Android devices [26]–[29]. Notwithstanding, Ebserver uses Android's ADB BATTERYSTATS to pool this data. BATTERYSTATS has also been used in other studies analysing energy consumption on mobile devices [47]–[49]. In regard to BATTERYSTATS accuracy, no statistical difference was found at 95% confidence when compared to physical devices [50].

Out of the 10 OWASP mobile vulnerabilities, this work focuses on four: M01, M02, M04, and M05. The experimental environment developed to test these points aims to cover commonly used scenarios where these vulnerabilities may manifest themselves. However, they are incomplete, and other scenarios to analyse these vulnerabilities may present somewhat different results. Nevertheless, our results show that increasing the security of an application will incur additional overhead in terms of energy. The remaining vulnerabilities presented by OWASP are more focused on the server side and software engineering aspects and can hardly be analysed by the experimental setup presented in this work. Nevertheless, developers should also consider these vulnerabilities when developing their applications.

VII. Conclusion and Future Works

This paper presents an analysis of the impact of energy consumption on the use of mobile device security solutions. The scope of this article is to investigate the security risks that mobile application developers can reproduce. We considered a list presented by OWASP of the top ten categories of mobile security risks to provide a comprehensive analysis of potential threats. Four risk categories fell within the scope of this article: improper platform usage, insecure data storage, insecure authentication, and insufficient cryptography.

An experiment was conducted to reproduce the energy consumption of operations, such as copying files and logging in, using a large set of operations with and without security. The results show that implementing encryption does have a statistically significant impact on the energy consumption of Android devices, thus answering **RQ1**. For the considered scenarios, the addition of encryption consistently leads to higher consumption, with the most extreme case showing a 1,000% increase. This increase appears to depend on the number of repetitions and the size of the encrypted data, which answers **RQ2**.

The results obtained from this research are noteworthy, as they point not only to an increase in energy consumption but also provide quantifiable data on this increase, providing scientific insight on the subject.

In future work, we intend to include metrics of memory and processing consumption in our analysis, in addition to energy consumption. It also expands our study's scope to include other security risk classifications, such as server-side manipulations and software engineering issues.

VIII. Acknowledgments

This work was financed by FEDER (Fundo Europeu de Desenvolvimento Regional), from the European Union through CENTRO 2020 (Programa Operacional Regional do Centro), under project CENTRO-01-0247-FEDER-047256 – GreenStamp: Mobile Energy Efficiency Services.

This work was financially supported by Base Funding - UIDB/00027/2020 of the Artificial Intelligence and Computer Science Laboratory – LIACC - funded by national funds through the FCT/MCTES (PIDDAC).

References

[1] H. Hoehle and V. Venkatesh, "Mobile application usability," *MIS quarterly*, vol. 39, no. 2, pp. 435–472, 2015.

[2] "How many smartphones are in the world?, howpublished = https://www.bankmycell.com/blog/how-many-phones-are-in-the-world# sources, note = Accessed: 2023-01-15."

[3] A. Rahmati, A. Qian, and L. Zhong, "Understanding human-battery interaction on mobile phones," in *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, 2007, pp. 265–272.

[4] "How to strengthen the security of your mobile applications to counter the most common attacks?" https://shorturl.at/qJM15, accessed: 2023-01-15.

[5] Y. Wang, K. Streff, and S. Raman, "Smartphone security challenges," *Computer*, vol. 45, no. 12, pp. 52–58, 2012.

[6] M. A. Hoque, M. Siekkinen, K. N. Khan, Y. Xiao, and S. Tarkoma, "Modeling, profiling, and debugging the energy consumption of mobile devices," *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, pp. 1–40, 2015.

[7] "Statcounter globalstats," https://gs.statcounter.com/os-market-share#monthly-202111-202210, accessed: 2023-01-14.

[8] D. Wu, G. D. Moody, J. Zhang, and P. B. Lowry, "Effects of the design of mobile security notifications and mobile app usability on users' security perceptions and continued use intention," *Information & Management*, vol. 57, no. 5, p. 103235, 2020.

[9] B. Watson and J. Zheng, "On the user awareness of mobile security recommendations," in *Proceedings of the SouthEast Conference*, 2017, pp. 120–127.

[10] C. Gonçalves, B. Sousa, and N. Antunes, "Oidc-tci: Oidc with trust context information," in *2022 14th IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, 2022, pp. 54–61.

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on February 25,2024 at 19:14:40 UTC from IEEE Xplore. Restrictions apply.

[11] "The open web application security project," https://owasp.org/www-project-mobile-top-10/, accessed: 2022-12-12.

[12] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback, "Report on the development of the advanced encryption standard (aes)," 2001-06-01 2001. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=151226

[13] M. J. Dworkin, "Recommendation for block cipher modes of operation: Methods and techniques," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., 2001. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-38A

[14] K. Moriarty, B. Kaliski, and A. Rusch, "Pkcs# 5: Password-based cryptography specification version 2.1," RFC Editor, RFC, January 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8018

[15] A. Lewerentz and J. Lindvall, "Performance and energy optimization for the android platform," 2012.

[16] "Your phone is almost out of battery. remain calm. call a doctor," https://www.wsj.com/articles/your-phone-is-almost-out-of-battery-remain-calm-call-a-doctor-1525449283, accessed: 2023-01-15.

[17] N. L. Bragazzi and G. Del Puente, "A proposal for including nomophobia in the new dsm-v," *Psychology research and behavior management*, vol. 7, p. 155, 2014.

[18] Z. Kholmatova, "Impact of programming languages on energy consumption for mobile devices," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 1693–1695. [Online]. Available: https://doi.org/10.1145/3368089.3418777

[19] A. McIntosh, S. Hassan, and A. Hindle, "What can android mobile app developers do about the energy consumption of machine learning?" *Empirical Software Engineering*, vol. 24, no. 2, pp. 562–601, 2019.

[20] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, M. D. Penta, R. Oliveto, and D. Poshyvanyk, "Multi-objective optimization of energy consumption of guis in android apps," *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 3, sep 2018. [Online]. Available: https://doi.org/10.1145/3241742

[21] L. Corral, A. Sillitti, and G. Succi, "Software assurance practices for mobile applications," *Computing*, vol. 97, no. 10, pp. 1001–1022, 2015.

[22] R. Abreu, M. Couto, L. Cruz, J. Cunha, J. P. Fernandes, R. Pereira, A. Perez, and J. Saraiva, "Green software lab: Towards an engineering discipline for green software," *arXiv preprint arXiv:2108.03028*, 2021.

[23] "Android batterystats," https://developer.android.com/topic/performance/power/setup-battery-historian, accessed: 2023-01-11.

[24] R. Rua and J. Saraiva, "E-manafa: Energy monitoring and analysis tool for android," in *37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–4.

[25] B. Moraes, W. Oliveira, F. Castor, and J. P. Fernandes, "Automating resource-usage data collection of android applications," IEEE/ACM 10th International Conference on Mobile Software Engineering and Systems (MOBILESoft) - Tools and Datasets, 2023.

[26] W. Oliveira, R. Oliveira, and F. Castor, "A Study on the Energy Consumption of Android App Development Approaches," in *MSR*, 2017.

[27] W. Oliveira, R. Oliveira, F. Castor, G. Pinto, and J. P. Fernandes, "Improving energy-efficiency by recommending java collections," *EMSE*, vol. 26, no. 3, p. 55, Apr 2021. [Online]. Available: https://doi.org/10.1007/s10664-021-09950-y

[28] C. Zimmerle, W. Oliveira, K. Gama, and F. Castor, "Reactive-based complex event processing: An overview and energy consumption analysis of cep.js," in *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, ser. SBES 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 84–93. [Online]. Available: https://doi.org/10.1145/3350768.3352492

[29] F. De Matos, W. Oliveira, F. Castor, P. Rego, and F. Trinta, "Multi-language offloading service: An android service aimed at mitigating the network consumption during computation offloading," in *Proceedings of the Brazilian Symposium on Multimedia and the Web*, ser. WebMedia '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 329–338. [Online]. Available: https://doi.org/10.1145/3539637.3557001

[30] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*. Berlin, Germany: Springer Publishing Company, Incorporated, 2012.

[31] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[32] R. D'Agostino and E. S. Pearson, "Tests for departure from normality. empirical results for the distributions of b2 and b1," *Biometrika*, vol. 60, no. 3, pp. 613–622, 1973.

[33] T. W. Anderson and D. A. Darling, "Asymptotic Theory of Certain "Goodness of Fit" Criteria Based on Stochastic Processes," *The Annals of Mathematical Statistics*, vol. 23, no. 2, pp. 193 – 212, 1952, publisher: Institute of Mathematical Statistics. [Online]. Available: https://doi.org/10.1214/aoms/1177729437

[34] I. Olkin, *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, ser. Stanford studies in mathematics and statistics. Palo Alto, CA: Stanford University Press, 1960.

[35] N. Weiss, P. Holmes, and M. Hardy, *A Course in Probability*. Upper Saddle River, NJ: Pearson Addison Wesley, 2005.

[36] A. Mazuera-Rozo, C. Escobar-Velásquez, J. Espitia-Acero, D. Vega-Guzmán, C. Trubiani, M. Linares-Vásquez, and G. Bavota, "Taxonomy of security weaknesses in java and kotlin android apps," *Journal of Systems and Software*, vol. 187, p. 111233, 2022.

[37] M. Tawalbeh, A. Eardley *et al.*, "Studying the energy consumption in mobile devices," *Procedia Computer Science*, vol. 94, pp. 183–189, 2016.

[38] N. Gregório, J. P. Fernandes, J. Bispo, and S. Medeiros, "E-apk: Energy pattern detection in decompiled android applications," in *Proceedings of the XXVI Brazilian Symposium on Programming Languages*, 2022, pp. 50–58.

[39] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2016.

[40] P. Weichbroth and Ł. Łysik, "Mobile security: Threats and best practices," *Mobile Information Systems*, vol. 2020, 2020.

[41] M.-L. Yao, M.-C. Chuang, and C.-C. Hsu, "The kano model analysis of features for mobile security applications," *Computers & Security*, vol. 78, pp. 336–346, 2018.

[42] J. A. Montenegro, M. Pinto, and L. Fuentes, "What do software developers need to know to build secure energy-efficient android applications?" *IEEE Access*, vol. 6, pp. 1428–1450, 2017.

[43] A. Merlo, M. Migliardi, P. Fontanelli, L. Spalazzi, and L. Viganò, "Measuring and estimating power consumption in android to support energy-based intrusion detection," *J. Comput. Secur.*, vol. 23, no. 5, p. 611–637, aug 2015. [Online]. Available: https://doi.org/10.3233/JCS-150530

[44] S. Huber, L. Demetz, and M. Felderer, "Pwa vs the others: A comparative study on the ui energy-efficiency of progressive web apps," in *Web Engineering*, 2021.

[45] M. Peters, G. . Scoccia, and I. Malavolta, "How does migrating to kotlin impact the run-time efficiency of android apps?" in *SCAM*, 2021, pp. 36–46.

[46] "Android kotlin," https://developer.android.com/kotlin, accessed: 2023-01-15.

[47] I. Malavolta, E. . Grua, C. Lam, R. de Vries, F. Tan, E. Zielinski, M. Peters, and L. Kaandorp, "A framework for the automatic execution of measurement-based experiments on android devices," in *ASE Workshops*, 2020, p. 61–66. [Online]. Available: https://doi.org/10.1145/3417113.3422184

[48] F. Bouaffar, O. L. Goaer, and A. Noureddine, "Powdroid: Energy profiling of android applications," in *ASEW Workshops*, 2021, pp. 251–254.

[49] S. Huber, L. Demetz, and M. Felderer, "A comparative study on the energy consumption of progressive web apps," *ISJ*, vol. 108, p. 102017, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437922000230

[50] D. Di Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, and A. De Lucia, "Software-based energy profiling of Android apps: Simple, efficient and reliable?" in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2017, pp. 103–114. [Online]. Available: http://ieeexplore.ieee.org/document/7884613/

Authorized licensed use limited to: b-on: Universidade de Coimbra. Downloaded on February 25,2024 at 19:14:40 UTC from IEEE Xplore. Restrictions apply.