Open Access
Full Length Article

ORIGINAL ARTICLE

# There is no such thing as free Lunch: An Investigation of Bloatware Effects on Smart Devices

**Iftikhar Alam[1], Muhammad Abid Khan[2], Muhammad Naeem[3], Shah Khusro[4]**

[1, 2, 3, 4, 5] Department of Computer Science, University of Peshawar, Peshawar

## ABSTRACT

**Introduction**: A software that provides minimal functionality at a cost of RAM and CPU cycles is considered as Bloatware. The explosion of bloatware in Android-based devices, such as smartphones and smart TVs, makes such devices deficient in terms of not only hardware-resources but also makes such devices more prone to security and privacy risks. Today's smartphones and smart TVs are equipped with powerful CPUs and a large amount of memory, making bloatware a least significant issue for the ordinary user. However, in reality, a user is paying in terms of battery life, unwanted space, compromise on privacy and security, and eventually cognitive overload.

**Objectives**: This paper presents a study of bloatware from security, privacy, energy, and storage perspectives. A categorization of the bloatware is proposed and some hazards of the bloatware problem for the smartphones system are identified.

**Methodology**: Some recommendations to deal with bloatware are made and an experimental study of some bloatware effects is presented. Furthermore, an empirical study is conducted to analyze user's perception about the pre-installed bloatware.

**Results**: Results show that bloatware significantly contributes to depletion of battery power. Furthermore, bloatware is a serious security issue and may capture private data of a user. It also contributes to degrading a device performance. Imperceptible types of bloatware, such as in-app ads, are also discussed. Besides security and performance issues, 97% users do not like in-app ads and hence methods of ads dispatch should be adapted.

**Conclusion**: This paper highlights overlooked issues related to bloatware in smart devices. In future, we intend to explore ways and means for minimizing the bloatware effect especially from software engineering point of view.

**Keywords:** Bloatware, Android-based devices, cognitive overload, smartphones

Author`s Contribution

*[1] Data analysis, interpretation and manuscript writing, Active participation in data collection*

*[2,3] Conception, synthesis, planning of research and manuscript writing*

*[4]Interpretation and discussion, Data Collection*

*Address of Correspondence*

*Shah Khusro*

*Email: khusro@upesh.edu.pk*

Article info.

*Received: Feb 06, 2017*

*Accepted: June 13, 2017*

***Cite this article:*** *Alam I, Abid Khan M, Naeem M, Khusro There is no such thing as free lunch: An investigation of bloatware effects on smart devices. J. inf. commun. technol. robot. appl.2017; 8(1):20-30.*

***Funding Source:*** *Nil*

***Conflict of Interest:*** *Nil*

## INTRODUCTION

The bloatware is pre-installed and often non-removable applications (apps in short) that come with smart devices like tablets, smartphones, and smart TVs [1, 2]. Also known as pre-installed bloats, they are installed by the companies or Original Equipment Manufacturer (OEM) for commercial reasons and incur a

cost on the customer [2, 3]. They have a high cost in terms of memory and CPU usage and battery consumption but provide only minimal functionality. For example, some studies report that 80% of the users use only 20% of the features of a bloatware but have to pay the full cost of the software in addition to paying the cost in terms of hardware overload as discussed above [4]. Addition or availability of more memory and processing power as in modern smartphone (See figures 1 and 2) doesn't provide sustainable solution [5-7] for two reasons: (i) additional hardware capabilities require more energy and energy is still a major concern for many smartphone consumers (ii) enhanced hardware capabilities increase the risk of battery drain [6, 8, 9]. The problem is aggravated by the fact that the bloatware in smartphone devices are stored in expensive primary memory instead of low-cost secondary memory as in the case of computers. Furthermore, smartphone devices have small display screens and bloatware may unnecessarily use screen space required by other applications. Last, bloatware may pose a security threat to user data [2, 10]. For example, some bloatware act as Trojan-horse [11], and can damage user's private data such as password, PIN code, contacts etc. Further, users usually prefer smartphones for storing their private data including passwords, bank account information, email, and pictures [12, 13]. As smartphones make extensive use of the internet, such data may be exposed to others especially to bloatware, which abounds on smartphone devices [14]. Moreover, most applications make a request for access-permission to unnecessary resources on a smartphone[13]. However, there is no mechanism available on smartphones to restrict an application from accessing unnecessary resources. Denying access permissions will not work because it restricts the user's ability to install new applications. In addition, many Android devices don't implement a sound access permission mechanism [15, 16] and this may increase the possibility of security and privacy failure in a smartphone[16].

In this paper, we undertake a study of the bloatware that involves a thorough literature review. We propose a categorization of bloatware that identifies types of applications that are bloatware or may behave as bloatware. Additionally, we also study the effects of bloatware on performance and resource use and identify some of these effects. Last, the study includes an experimental module where we investigate the effects of some of the bloatware. Rest of the paper is organized as follows. Section II is related work. Section III gives a classification of bloatware with experimental results. Section IV includes recommendations for companies and developers for dealing with bloatware and identifies some directions for research. Sections V and VI present an empirical study for finding a user' perception about bloatware. Section VII gives conclusion and goals for future work.

## LITERATURE REVIEW

As per definition in [4] and discussion in introduction section, the bloatware is not limited to unwanted apps only. The bloatware effect may be caused by a number of other factors such as new releases and upgrades by companies and developers to their software that often result in more resource-hungry applications, malware, and large amount of data generated by sensors attached to modern smartphones. Poor compliance with software engineering standards for mobile applications may result in applications that are not optimized for efficient usage of hardware resources. Likewise, battery power depletion and issues related to privacy & security of personal data in smartphones are not the only problems caused by bloatware. Other problems caused by bloatware are performance degradation, wastage of screen space, and low rating for companies when users assign low rates to products they perceive as unwanted or intrusive/disruptive. Besides, bloatware may also cause direct financial cost to the customer.

The little attention has been paid to the problem of bloatware so far. Although hardware capabilities of the smartphone have seen constant growth (See Figures 1 and 2), that does not solve the problem because bloatware equally grows in number and size. A few measures that have been proposed (to the best of our knowledge) include the detection of resource-hungry applications and reporting them to the user. Similarly, tools for detecting and removing malware or blocking adware abound yet the problem of bloatware has not been satisfactorily handled. Most importantly, the problem of bloatware has not been studied and understood

thoroughly.

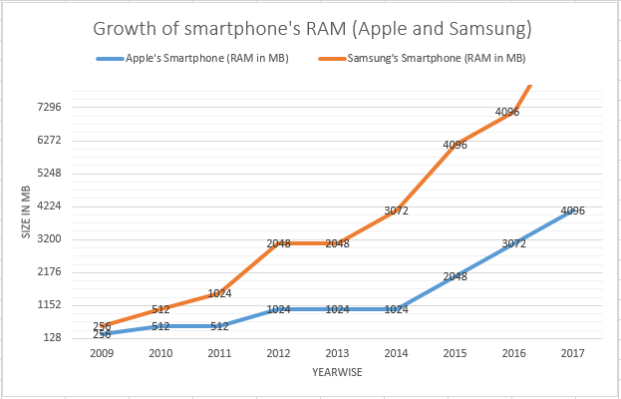


**Fig. 1. RAM growth in megabytes (Data Compiled from Wikipedia)**

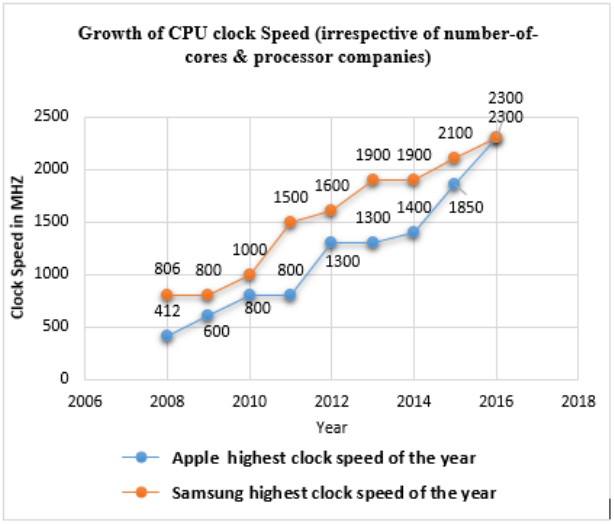


**Fig. 2. Clock Speed growth (Data compiled from Wikipedia)**

## CLASSIFICATION OF BLOATWARE:

Through our literature review, we identified a variety of causes for the bloatware effect. These range from unwanted pre-installed applications to software upgrades and new releases, rapid updates, etc. Evidence for some of the causes was obtained through the experiments in Section VI. In this section, we give a classification of bloatware based on the causes. In addition, we also identify the effects of the bloatware. Table 1 is for the various types of bloatware. The rest of the content describes different types of detail.

Classification of bloatware and its negative consequences are described in Table 1. The most widely discussed type in literature is "unwanted apps". Other types that consume precious resources without significant benefits including creeping featurism, unwanted sensory data, quickly & dirty developed applications, malicious applications, In-app ads, and rapid updates. Companies and developers have a variety of reasons for producing bloatware. The most common reason is commercial. For example, the purpose of producing pre-installed apps (bloatware) is "extra revenue" to companies. Other reasons include software enrichment, Special purpose applications, Quick development & launching of applications, Market dominance, User's demands, Updated apps and Negative intentions. The effects that bloatware causes are a low rating, wastage of smartphone's RAM/ROM, wastage of smartphone's CPU cycles, wastage of smartphone's screen space, security issues, extra data charges due to rapid updates, low-quality apps, and unreliable apps. The content below in Table 1 gives a description of these types.

### A. Unwanted apps

The most discussed bloatware in literature is the unwanted software that is supplied with a device for generating additional revenue for firms and companies [2, 10, 17]. Android-based devices are especially the target of such practice because of the popularity of Android operating systems [3]. Usually, the unwanted applications receive negative comments and low rating from users. However, not all applications with a low rating are bloatware. Special-purpose applications also usually receive low ratings. Most of the unwanted applications running in the background (in RAM) consuming extra CPU cycles and causing the depletion of battery power. Additionally, they may remain in the system undetected for a long time [1]. Another disadvantage of unwanted applications is that most of them involve a cost and thus add to the cost of the device.

| Table 1: Classification of bloatware | | |
|---|---|---|
| Types of bloatware | Purpose/intentions | Negative consequences |
| **Unwanted apps** | Revenue Market dominance | Low rates |
| | | Wastage of RAM/Rom |
| | | Wastage of CPU cycles |
| | | Security issues |
| | | Energy consumption |
| | | Low quality |
| **Creeping featurism** | Revenue Enrichment | Wastage of CPU cycles |
| | | Wastage of RAM/ROM |
| **Dirty apps** | Quick launch Market dominance | Wastage of RAM/Rom |
| | | Wastage of CPU cycles |
| | | Energy consumption |
| **Malware apps** | Negative intentions | Low rates |
| | | Low quality |
| | | Security issues |
| **In-app ads** | Revenue | Low rates |
| | | Energy consumption |
| | | Security issues |
| | | Wastage of RAM |
| **Rapid updates** | Bugs fixation Enrichments | Wastage of RAM/ROM |
| | | Wastage of CPU cycles |
| | | Energy consumption |

Table-1 shows the types, purpose, negative consequences, and literature in which it is discussed. A machine learning technique can be used to distinguish between a useful app and a bloatware. A machine (smartphone after some time will learn and will automatically detect the possible bloatware. For testing purposed, we chose usage frequency as the test to distinguish unwanted (bloatware) applications from useful applications, i.e., more frequently accessed applications were defined as useful and not accessed or minimally accessed as bloatware. For experimental purposes, we selected ten Android phone users and studied each user's usage pattern. To gather usage statistics, App-Usage was used as the tool [18]. For this purpose, applications on each smartphone were grouped into system applications and installed applications using App-usage and a count of applications in each category was obtained. The number of total apps, system apps, and installed apps is given in Table 2.

The applications that were accessed over a period of one month were counted. The number of installed applications that were never accessed during the one-month period yielded the possible number of bloatware per device as shown in Table 2. As can be seen, the average number of possible bloatware per device is 12.3. This is a significant number not to mention that the bloatware so identified may include large-sized hardware-hungry applications.

| Android Phones | Total apps | System apps | Installed Apps | Possible bloatware (Never accessed) |
|---|---|---|---|---|
| 1 | 58 | 23 | 35 | 12 |
| 2 | 53 | 22 | 31 | 10 |
| 3 | 57 | 24 | 33 | 10 |
| 4 | 52 | 21 | 31 | 11 |
| 5 | 60 | 24 | 36 | 12 |
| 6 | 52 | 21 | 31 | 13 |
| 7 | 62 | 23 | 39 | 17 |
| 8 | 54 | 22 | 32 | 11 |
| 9 | 57 | 23 | 34 | 14 |
| 10 | 61 | 24 | 37 | 13 |
| Average | 56.6 | 22.7 | 33.9 | 12.3 |

### B. Creeping featurism

Addition of more features to products and releasing new versions is an integral characteristic of the working of the smartphone industry. Often users also like feature-rich products and are willing to pay the cost albeit without wanting the bloatware [19]. Upgradation/enhancement to applications, however, adds to the size and complexity of software often without any significant benefit to the user [20]. A consequence is a need for more resources especially in terms of RAM and battery power. This makes an apparently useful application to degrade bloatware[21, 22]. An example is the Samsung (smartphone) version S4 release with unwanted applications and additional system files that consume up to 45% of the internal storage space[2].

An experiment is conducted for analyzing the size growth of an app, we downloaded 100 top-rated free applications with a maximum size of 64 MB from Google-Play store. The experiments were conducted on both the system and installed applications (applications that are installed by smartphone manufacturer or Original Equipment Manufacturer). For analyzing size-growth and complexity, updates were applied to the applications, the size was found to grow by a large amount for all applications except the top 20 for which updates were not available. See Table 3 for average size (in MB) of each application before and after updates.

It should be noted that we were able to analyze only a few popular applications. It needs to be mentioned that most applications require regular updates. Additionally,

updates occur throughout the lifetime of applications.

| Table 3: Size in MB (before & after updates) | | | |
|---|---|---|---|
| **Application** | **Average Size (before updates)** | **Average Size (after updates)** | **Increase in %** |
| Gmail | 12.22 | 39.98 | 227.16% |
| Play Store | 21.3 | 40.70 | 91.07% |
| Maps | 33.1 | 121.0 | 265.55% |
| Music | 7.0 | 19.09 | 172.71% |
| S Health | 30.41 | 168.0 | 452.44% |
| Galaxy Apps | 8.01 | 68.55 | 755.80% |
| **Total** | **112.4 MB** | **457.32 MB** | **327.45% (Average increase)** |

The same procedure was applied to 100 application that was installed for analysis. Overall, we observed almost 10% increase in the size of an application in a month period. The increase in the size of 100 apps has been shown in Fig 3.

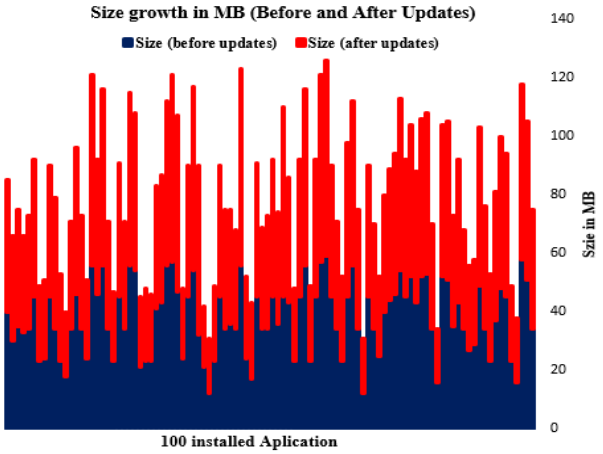


**Fig. 3. Size of applications (Before & After updates)**

## C. Unwanted Sensory Data

Most of the modern smartphones and smartwatches are equipped with sensors that generate a large amount of data on daily basis [7, 23, 24]. The availability of such data provides opportunities to developers for developing specialized mobile applications [25]. However, handling such data requires additional CPU time and battery power [25-27]. The user does not require most of the sensor data. Moreover, common smartphone users rarely use sensory data. Thus, the presence of the unwanted sensory data in the system may give rise to bloatware.

An experimental result of power consumption in milliamp (mA) on Samsung SM-N9005 with Android 5.0 is shown in Figure 4. The application [28] installed on our above mentioned testbed device shows a complete list of the available sensor and their power consumption in milliamp (mA). The sensors and power consumption is visualized in Figure 4.

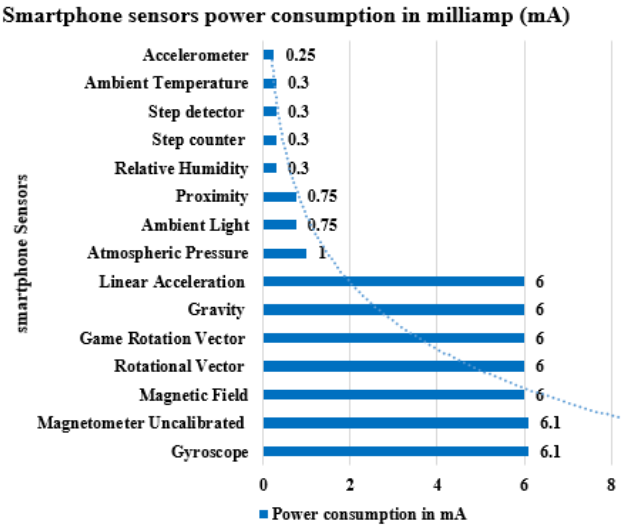


**Fig. 4. Sensor power consumption**

Sensory data is not always utilized for every individual. A user who use their smartphones only for communication and basic operations, as mostly practice in developing countries and by non-technical peoples, become the victims.

## D. Quick and Dirty Development

The rapid development of applications with modern software engineering tools may cause bloatware [21]. This is because applications may be developed without proper planning and compliance with standard engineering processes, e.g., sound testing of the application may be ignored [29]. Further, rapid application development usually involves building the software by combing different pieces of code in the form of APIs, packages, and classes. Consequently, little attention may be paid to optimize the product with regard to resource utilization such as CPU usage and efficient/economical use of battery power.

## E. Malware (malicious applications)

Malware is bloatware. In addition to stealing CPU cycles and battery-power from the system, they also pose a threat to user data [30]. A common reason for the presence of malware in smartphone devices is downloading software from an unsafe source such as an

application-store (app-store) that contains malware. Another reason is the existence of ad-libraries (advertisement libraries) in the system. Both of these applications may behave as bloatware. Additionally, they may steal personal data and download malicious code from remote servers[31]. As an open source platform, Android operating system is more vulnerable to malicious apps intrusion[32, 33].

We also performed access permission analysis of the test applications. For this purpose, a few popular permission parameters were chosen. Results are presented in Table 4.

**Table 4: Android Permission**

| Permission parameters | % of apps allowed |
|---|---|
| Modify or delete the contents of your SD card | 98% |
| Read the contents of your SD card | 98% |
| Add or remove accounts | 68% |
| Read your contacts | 60% |
| Modify your contacts | 60% |
| Read phone status and identity | 54% |

As can be seen, 98% of the free apps have "dangerous" permission right granted by the android system. When access permissions for individual applications were analyzed, 98% of the applications were found to have read, modify, or delete permission for the contents of the SD card. This means a malicious application can access the memory card of a smartphone and steal or delete private data of a user. Further, among the 100 applications, 68% applications could add or remove user accounts. Additionally, 60% of applications could access both public and private contacts on the smartphone device. Finally, 54% of the applications could read phone status-and-identity of users.

### F. In-app ads (Adware)

Advertisement (ads) is a major source of income for developers, firms, and companies. Applications embedded with bundles of advertisements are considered adware. Adware is considered a kind of bloatware although not all adware is bloatware [34]. Moreover, adware can be a security threat especially when it downloads and run malicious code from remote servers [31]. Technically, any mobile application equipped with high severity ad-libraries is termed as Madware [30]. Madware can be embedded in the app and can easily spread [35]. Applications available on non-authentic source should be avoided for security reasons [36].

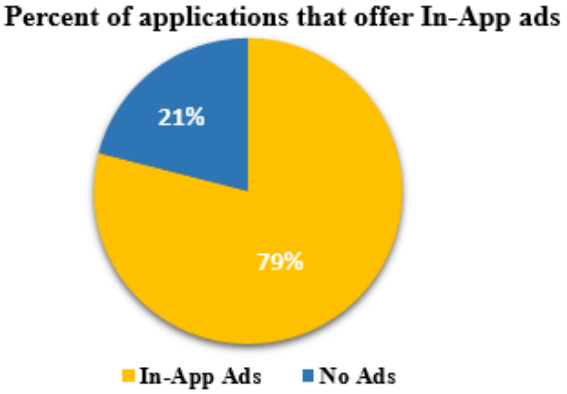


**Fig. 5. In-app ads**

Users do not like an in-app advertisement [37]. As per the definition of bloatware, the advertisement can also consume memory and CPU cycles without a perceivable benefit to the user. Additionally, advertisement libraries (ad-libraries) are a security threat. We downloaded some free applications and found that 79% were accompanied by bundles of advertisements for commercial purposes. See Figure 5 for the findings.

### G. Rapid Updates

Updates play a major role in maintaining stability and security of a device [38, 39]. Smartphone's applications are frequently updated to fix bugs, provide enhancements, and ensure security and maintenance [40]. However, an issue with updating services is that they need to be in the RAM to apply updates and to synchronize and update user accounts [41]. Their behavior may thus degenerate to that of a bloatware if the updates are too frequent without any tangible benefit. An example is the Google Play, which provides an easy way of deploying new updates to their applications. However, poor quality assurance standards of applications on Google Play store can lead to excessive updates for trivial benefits [42].

In our experiment, we also examined the frequency of updates to applications. We found that applications in Games, Beauty, and Education categories are updated on regular basis. In addition, two to three updates (mean= 3.5, Standard Deviation (SD) = 0.57) are done on daily basis.

## RESEARCH OPPORTUNITIES

So far, few solutions have been proposed to the problem of bloatware. Adding more memory or battery power is not an effective solution in addition to being costly. Recently some tools for detecting energy-hungry applications have been proposed. TIDE[43] is one such tool for Android-based smartphones. However, only identification of energy-hungry applications is not a viable solution to cope with security and power constraints on smartphones. Another solution would be adjusting a smartphone device towards power-saving. For example, in [26, 44], a power management framework has been proposed for Android-based devices. In the framework, if a user in the non-active group falls in a high-power consumption group, that is considered an abnormal usage behavior and is notified to the user [44].

Similarly, the problem of malicious applications on a smartphone has also not been effectively tackled. Although, some smartphones have more robust security systems that don't allow downloading applications from unsafe sources, the more common android is an open source operating system that permits installation of software from any source. Security mechanisms on Android, e.g., sandbox, signature, and permission, are ineffective because the permission process is usually casual if the user is unaware of security and privacy of data [15, 16]. Moreover, there is no way to restrict the permissions that an application requests. Some bloatware exploits the permission process to steal and damage user data [45, 46]. The code snippet in Fig. 4 from android developers1 characterizes the potential risk embedded in the permission. The low-risk permission level "normal" is automatically granted by the system. However, granting the high-risk permission level "dangerous" can allow access to private data or a control on a smartphone device.

```
<permission android:description="string resource"
            android:icon="drawable resource"
            android:label="string resource"
            android:name="string"
            android:permissionGroup="string"
            android:protectionLevel=["normal" | "dangerous" |
                                     "signature" | "signatureOrSystem"] />
```

**Fig. 6. Code snippet for android permission**

[1] https://developer.android.com/guide/topics/manifest/permission-element.html, Access date: 23 September 2017

As highlighted in the foregoing discussion, the problem of bloatware has not received adequate attention from software developers and companies. Methods proposed for the detection of energy-hungry applications are not very effective in identifying bloatware. In our view, companies need to design products judiciously and optimize them with regard to the usage of system resources to avoid the bloatware problem. Detection of applications that are not used by the user or development of more efficient methods for detecting energy-hungry applications are some areas that need the attention of researchers and developers. Below, we recommend some strategies for tackling the problem of bloatware.

### A. Dealing with Creeping Featurism

Creeping featurism is the propensity in companies to make their products more attractive by adding new features and make enhancements. If done in an unsystematic way, enhancements may result in problems such as larger application size, security issues, code complexity, and unreliability [21, 47]. A consequence of this may be bloatware effect. Another consequence may be low user-friendliness especially for a senior citizen, users with disabilities, and non-technical users. Furthermore, rich features in particular less-user-friendly interfaces on relatively smaller screens of smartphones may cause the cognitive overload problem for some users [48]. We think that bloatware problem can be mitigated by adapting/tuning the addition of new features to user needs. In particular, the trade-off between usability and functionality should be taken into account especially for different age groups and regions when adding features to applications. For example, for elderly, novice, and casual users, basic software functionality may be considered enough [22]. Last, developers should try to keep the next version more stable, more user-friendly, and less hardware hungry. Maximizing backward compatibility and reducing cognitive overload for users may minimize bloatware effect and enhance the efficiency of applications.

### B. Software Engineering Tools and Techniques

Some bloatware is the results of poor compliance of mobile software with software engineering standards (or the lack of proper engineering standards for mobile applications), especially improper attention to non-functional requirements [49, 50], such as requirements

related to software power consumption, security, and performance. Moreover, traditional software engineering techniques for desktop development may not be effective in the domain of smartphone application development [51]. This is because smartphones have diverse hardware and operating systems that may pose specific challenges to the developer. Therefore, traditional techniques may be practiced with care. Further, long-term use/deployment of applications in the case of the smartphone may not be effective as hardware and operating system features of smartphones are continuously evolving.

### C. Utilization of sensory data

Continuous generation of data by sensing devices installed in many smartphones is expensive because the transportation of sensor data to system-on-chip requires the constant attention of the CPU [7]. Furthermore, a large amount of sensory data may cause energy insufficiency problems on Android-based phones. So far, little attention has been paid to the problem of bulk sensory data on smartphones. Some approaches based on the detection and analyses of the utilization of sensor data have been proposed in [25, 26, 44]. However, there is room for further research in particular with regard to finding better ways for the utilization of sensor data. An approach to tackle the issue of unwanted sensory data is to make sensors generate data on demand (by applications). Another potential solution may be to take regional trends into account when developing mobile applications as users of different regions has different requirements for mobile applications and may need simple smartphone features [52].

### D. Controlling pre-installed bloatware

Due to market competition, mobile phone companies are usually under immense pressure to release better and feature-rich smartphones in terms of both hardware and software. However, companies do not carefully assess the usefulness of these applications for consumers and as a consequence, many new applications degrade bloatware [2]. Companies should avoid pre-installation of software if there is the frequent removal of such software by customers. An example is the NEXUS series by Google, which supports direct rooting or quick rooting with a few simple steps [2]. An issue with these techniques, however, is that privacy and security may be compromised [1, 53].

### E. On-demand updates

Operating system and application updates play a major role in maintaining the stability and security of a device [38, 39, 54]. However, lengthy and insecure updating process can damage a device and data [36]. Moreover, rapid updates are not always welcome by the smartphone users [55], especially when most features or User Interface (UI) are changed significantly by the updates [54]. In addition, it may waste online data usage [2]. Excessive updates may degrade a mobile application to bloatware. A potential solution can be updated on demand instead of applying updates automatically. Moreover, as user behavior to updates differs significantly across the countries, companies and developers should take the country of their target users into consideration when choosing updates [52].

## METHODS AND MATERIAL

Beside some experimental results, an empirical study was conducted to investigate the user's perception about bloatware. We conducted a mixed mode survey (observation, interview, and questionnaire) for analysis. The type of interview was structured. A random sample of 350 smartphone's users was surveyed at the university. These include students, teachers, and professionals. There were 23 questions in the survey, which 03 were personal information and were not included in results and analysis. We tried the elicitation of user' perception about bloatware from these perspectives (1) about unwanted apps (2) about rapid growth (3) about consequences in terms of unwanted power consumption (4) about rapid updates (5) about security & privacy concerns specifically from bloatware perspectives and (6) about in-app ads.

### A. Demographics

A random sample of 350 respondents was selected for analysis. To make the survey unbiased, we tried to survey every age group including male and female. The following table shows the demography of participants/respondents.

Table 5. Demographics

| Table 5. Demographics | | | |
|---|---|---|---|
| Participants | Demographics | Number of Participants | Percentage |
| Gender | Female | 60 | 17.14 |
| | Male | 290 | 82.85 |
| Age group | 20 to 30 Years | 120 | 34.28 |
| | 31 to 40 Years | 110 | 31.42 |
| | 41 to 50 Years | 80 | 22.85 |
| | others | 40 | 11.42 |
| Background | Educated | 280 | 80 |
| | Literate | 70 | 20 |
| Smartphone Usage Experience | 1 years | 60 | 17.14 |
| | 2 year | 80 | 22.85 |
| | 3 year | 80 | 22.85 |
| | 4 years | 65 | 18.57 |
| | 5 years | 55 | 15.71 |
| | others | 10 | 2.85 |

## RESULTS AND ANAYSIS

### A. Smartphone's Operating system

Observing user's smartphones, we found that 80% of users have an Android operating system. One reason for this high number of Android user was low-price affordable smartphones in the surveyed region. Apple's iOS was installed on 7% of the smartphone. Rests of 13% users have another operating system including Windows and legacy Nokia's Symbian operating system.

### B. Investigating Unwanted apps

By estimating the number of unwanted apps in a smartphone, we first educated the respondent that what we mean by the unwanted app and then collected the responses using interviews and observation. By interviewing and observing a user's smartphone, we found that 90% of the total apps were pre-installed on their smartphones. We found that on average 10% of apps were installed. Except for basic telephony, i.e. SMS, Call, Email etc., rest of the pre-installed apps were rarely used.

### C. Analysing the rapid growth and size

Analyzing the growth and size of pre-installed apps, we found that 60% of the heavyweight application was pre-installed and declared as bloatware due to their rear usage. These heavyweight applications become more obese after major updates. We also found that 30% of smartphones users were suffering from low memory space due to the large application. Because of rare usage, 72 % of people want to uninstall pre-installed heavyweight apps. However, 90% of users were not aware of rooting, jailbreaking, and their negative consequences. In addition, these heavyweight applications clutter the smartphone's user interface and contribute to cognitive overload especially for non-technical and senior citizens.

### D. Unwanted power consumption by sensors

Surprisingly, 82% of smartphone's users do not track the RAM usage nor they clear their smartphone's cache. In addition, 95% of the people do not close their running apps in memory. Furthermore, 91% of the people were not aware of unnecessary power consumption of smartphone's sensors. Except for games, which use the accelerometer, gyros etc., 80% of people have no application for proper usage of sensory data and hence a wastage of continues power consumption by the smartphone's sensors.

### E. Examining Rapid Updates

By examining responses, we found that 60% of people do not update their apps. Only 16 % of users have updated their OS. Although, a few users wish to update their smartphone operating system, due to unavailability of an updated version for their smartphones restrict them to older versions. Only 25% of people update overall apps, while 30% of people update only their desired apps. Analyzing user responses, we found that people do not like regular updates nor they like regular prompts about updates.

### F. Malicious Apps and Android Permissions

Installing apps and games on a smartphone is a normal behavior. However, 93% of people are not aware of Android permissions that are requested by an app during installation. They never read in detail the permissions popups and usually grating these permissions to an app is casual. We found that 98% of the people were not aware of negative consequences of blindly grating the permissions to an app Furthermore; we found that 85% of people have installed their apps from authentic source i.e. Google's Play and Apple's iTunes. Only 15 % of people have installed their apps from other sources including sharing of Android Package (.APK) files.

### G. In-app ads and security concerns

By observing and analyzing the collected responses, we found that users prefer free apps. However, 97% of the people do not like in-app ads that come with free apps. One major reason for disliking in-app ads is that it ucts a

## CONCLUSION AND FUTURE WORK

In this paper, we explored the problem of bloatware and the effects on device resources and security and privacy of user data. Through our experimental and empirical study, we found that in addition to manifesting in the form of pre-installed software, bloatware effect is caused by a number of other practices/factors such as excessive featurism, software upgrades and new releases, rapid application updates, and bulky sensor data. We found a good evidence of the relationship between the size and complexity of applications and the bloatware effect. The connection underlined the need for a tradeoff between software quality and functionality, and consumption of device resources especially the constraints on the battery power and the RAM and CPU usage. Nowadays, developers are not giving due attention to save bits, bytes and processor cycles for the development of software applications2. In our view, a potential solution to the problem of bloatware may be the proper planning of the software design process, better compliance with software engineering standards, and more focus on the real needs and expectations of users. In particular, component-based assembly, and rapid development should be geared towards optimal resource usage, security, and privacy. In future, we intend to explore ways and means for minimizing the bloatware effect especially from software engineering point of view.

## REFERENCES


1. Vrhovec, S. L. R. (2016, May). Safe use of mobile devices in the cyberspace. In Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2016 39th International Convention on (pp. 1397-1401). IEEE.
2. H. Cavusoglu, H. Cavusoglu, and X. Geng, "Bloatware and Jailbreaking: How Consumer-Initiated Modification Interacts with Product Pricing," 2016.
3. McDaniel, P. (2012). Bloatware comes to the smartphone. IEEE Security & Privacy, 10(4), 85-87.
4. Spolsky, J. (2004). Strategy Letter IV: Bloatware and the 80/20 myth. In Joel on Software (pp. 277-280). Apress, Berkeley, CA.
5. Picco, G. P., Julien, C., Murphy, A. L., Musolesi, M., & Roman, G. C. (2014, May). Software engineering for mobility: reflecting on the past, peering into the future. In Proceedings of the on Future of Software Engineering (pp. 13-28). ACM.
6. Ali, F. A., Simoens, P., Verbelen, T., Demeester, P., & Dhoedt, B. (2016). Mobile device power models for energy efficient dynamic offloading at runtime. Journal of Systems and Software, 113, 173-187.
7. Fan, S., Llull, Q., & Lee, B. C. (2017, February). Predicting Sensory Data and Extending Battery Life for Wearable Devices. In Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications (pp. 43-48). ACM.
8. Kim, S. H., Jeong, J., Kim, J. S., & Maeng, S. (2016). SmartLink: A memory reclamation scheme for improving user-perceived app launch time. ACM Transactions on Embedded Computing Systems (TECS), 15(3), 47.
9. Abou-Of, M. A., Taha, A. H., & Sedky, A. A. (2016, April). The trade-off between low power and energy efficiency in benchmarking. In Information and Communication Systems (ICICS), 2016 7th International Conference on (pp. 322-326). IEEE.
10. Elahi, H., Wang, G., & Li, X. (2017, December). Smartphone Bloatware: An Overlooked Privacy Problem. In International Conference on Security, Privacy and Anonymity in Computation, Communication, and Storage (pp. 169-185). Springer, Cham.
11. Wikipedia. (2017, 25 March). Trojan Horse. Available: https://en.wikipedia.org/wiki/Trojan_horse_(computing)
12. Huang, Y., Chapman, P., & Evans, D. (2011, August). Privacy-Preserving Applications on Smartphones. In HotSec.
13. Zhang, L. L., Liang, C. J. M., Li, Z. L., Liu, Y., Zhao, F., & Chen, E. (2018). Characterizing privacy risks of mobile apps with sensitivity analysis. IEEE Transactions on Mobile Computing, 17(2), 279-292.
14. Finkelstein, A., Biton, R., Puzis, R., & Shabtai, A. (2017). Classification of smartphone users using internet traffic. arXiv preprint arXiv:1701.00220.
15. Singh, P., Tiwari, P., & Singh, S. (2016). Analysis of the malicious behavior of Android apps. Procedia Computer Science, 79, 215-220.
16. Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., & Wagner, D. (2012, July). Android permissions: User attention, comprehension, and behavior. In Proceedings of the eighth symposium on usable privacy and security (p. 3). ACM.
17. Cavusoglu, H., Cavusoglu, H., & Geng, X. (2015). Should Firms Bundle Bloatware with Consumer Electronics?–Implications for Product Pricing and Consumer Surplus.
18. Löfvenmark, K. (2017). The Constant Connectivity Conundrum: Experiences of multitasking and interruptions.
19. Hamilton, J. (1999). Fault Avoidance vs. Fault Tolerance: Testing Doesn't Scale. In High-Performance Transaction Systems (HPTS) Workshop.
20. Bagheri, H., Garcia, J., Sadeghi, A., Malek, S., & Medvidovic, N. (2016). Software architectural principles in contemporary mobile software: from conception to practice. Journal of Systems and Software, 119, 31-44.
21. Quach, A., Prakash, A., & Yan, L. K. (2018). Debloating Software through Piece-Wise Compilation and Loading. arXiv preprint arXiv:1802.00759.
22. Jiang, Y., Zhang, C., Wu, D., & Liu, P. (2015, August). A preliminary analysis and case study of feature-based software customization. In Software Quality, Reliability and Security-


[2] https://en.wikipedia.org/wiki/Software_bloat

Companion (QRS-C), 2015 IEEE International Conference on (pp. 184-185). IEEE.
23. Wu, Y., & Slyke, C. V. (2005). Interface complexity and elderly users: Revisited.
24. Sandstrom, G. M., Lathia, N., Mascolo, C., & Rentfrow, P. J. (2016). Opportunities for smartphones in clinical care: the future of mobile mood monitoring. The Journal of clinical psychiatry, 77(2), e135.
25. Otebolaku, A. M., & Andrade, M. T. (2016). User context recognition using smartphone sensors and classification models. Journal of Network and Computer Applications, 66, 33-51.
26. Liu, Y., Xu, C., & Cheung, S. C. (2013, March). Where has my battery gone? Finding sensor related energy black holes in smartphone applications. In Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on (pp. 2-10). IEEE.
27. Li, Q., Xu, C., Liu, Y., Cao, C., Ma, X., & Lü, J. (2017). CyanDroid: stable and effective energy inefficiency diagnosis for Android apps. Science China Information Sciences, 60(1), 012104.
28. Li, H., Liu, X., & Mei, Q. (2018). Predicting Smartphone Battery Life based on Comprehensive and Real-time Usage Data. arXiv preprint arXiv:1801.04069.
29. Kwiatkowski.mobi, "Sensor List," 2013.
30. Deng, L., Offutt, J., Ammann, P., & Mirzaei, N. (2017). Mutation operators for testing Android apps. Information and Software Technology, 81, 154-168.
31. Uscilowski, B. (2013). Mobile adware and malware analysis. Symantec Corp, 1.
32. Grace, M. C., Zhou, W., Jiang, X., & Sadeghi, A. R. (2012, April). Unsafe exposure analysis of mobile in-app advertisements. In Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks (pp. 101-112). ACM.
33. Zhu, H. J., You, Z. H., Zhu, Z. X., Shi, W. L., Chen, X., & Cheng, L. (2018). DroidDet: Effective and robust detection of Android malware using static analysis along with rotation forest model. Neurocomputing, 272, 638-646.
34. BalaGanesh, D., Chakrabarti, A., & Midhunchakkaravarthy, D. (2018). Smart devises Threats, Vulnerabilities and Malware Detection Approaches A Survey. European Journal of Engineering Research and Science, 3(2), 7-12.
35. Haddadi, H., Guha, S., & Francis, P. (2009, September). Not all adware is badware: Towards privacy-aware advertising. In Conference on e-Business, e-Services and e-Society (pp. 161-172). Springer, Berlin, Heidelberg.
36. Rastogi, S., Bhushan, K., & Gupta, B. B. (2016). Measuring Android App Repackaging Prevalence based on the Permissions of App. Procedia Technology, 24, 1436-1444.
37. Xu, M., Song, C., Ji, Y., Shih, M. W., Lu, K., Zheng, C., ... & Lee, S. (2016). Toward engineering a secure Android ecosystem: A survey of existing techniques. ACM Computing Surveys (CSUR), 49(2), 38.
38. Gui, J., Nagappan, M., & Halfond, W. G. (2017). What Aspects of Mobile Ads Do Users Care About? An Empirical Study of Mobile In-app Ad Reviews. arXiv preprint arXiv:1702.07681.
39. Vaniea, K., & Rashidi, Y. (2016, May). Tales of software updates: The process of updating software. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (pp. 3215-3226). ACM.
40. Wash, R., Rader, E., Vaniea, K., & Rizor, M. (2014, July). Out of the loop: How automated software updates cause unintended security consequences. In Symposium on Usable Privacy and Security (SOUPS) (pp. 89-104).
41. McIlroy, S., Ali, N., & Hassan, A. E. (2016). Fresh apps: an empirical study of frequently-updated mobile apps in the Google play store. Empirical Software Engineering, 21(3), 1346-1370.
42. J. Wallen. (2015, 25 March). 3 reasons Google Play Services might be draining your battery [Blog]. Available: http://www.techrepublic.com/article/3-reasons-google-play-services-might-be-draining-your-battery/
43. Comino, S., Manenti, F. M., & Mariuzzo, F. (2016, November). To upgrade or not to upgrade? the release of new versions to survive in the hypercompetitive app market. In Proceedings of the International Workshop on App Market Analytics (pp. 37-42). ACM.
44. Dao, T. A., Singh, I., Madhyastha, H. V., Krishnamurthy, S. V., Cao, G., & Mohapatra, P. (2017). TIDE: A user-centric tool for identifying energy-hungry applications on smartphones. IEEE/ACM Transactions on Networking, 25(3), 1459-1474.
45. Duan, L. T., Lawo, M., Rügge, I., & Yu, X. (2017). Power Management of Smartphones Based on Device Usage Patterns. In Dynamics in Logistics (pp. 197-207). Springer, Cham.
46. Mandaliya, V., Jain, B., Palekar, S., & Patel, K. (2016). Permission Based Risk Communication For Android Apps.
47. Karim, M. Y., Kagdi, H., & Di Penta, M. (2016, March). Mining android apps to recommend permissions. In Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on (Vol. 1, pp. 427-437). IEEE.
48. McGrenere, J. (2000, April). Bloat: the objective and subject dimensions. In CHI'00 Extended Abstracts on Human Factors in Computing Systems (pp. 337-338). ACM.
49. Kurniawan, S. (2008). Older people and mobile phones: A multi-method investigation. International Journal of Human-Computer Studies, 66(12), 889-901.
50. Barn, R., & Barn, B. S. (2016, February). Integrating Values into Mobile Software Engineering. In Proceedings of the 9th India Software Engineering Conference (pp. 196-196). ACM.
51. Nagappan, M., & Shihab, E. (2016, March). Future trends in software engineering research for mobile apps. In Software analysis, evolution, and reengineering (SANER), 2016 IEEE 23rd International Conference on (Vol. 5, pp. 21-32). IEEE.
52. Kumar, N. A., Krishna, K. H., & Manjula, R. (2016). Challenges and best practices in mobile application development. Imperial Journal of Interdisciplinary Research, 2(12).
53. Lim, S. L., Bentley, P. J., Kanakam, N., Ishikawa, F., & Honiden, S. (2015). Investigating country differences in mobile app user behavior and challenges for software engineering. IEEE Transactions on Software Engineering, 41(1), 40-64.
54. Zhang, H., She, D., & Qian, Z. (2015, October). Android root and its providers: A double-edged sword. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 1093-1104). ACM.
55. Dodier-Lazaro, S., Becker, I., Krinke, J., & Sasse, M. (2017). No Good Reason to Remove Features: Expert Users Value Useful Apps over Secure Ones.
56. Tian, Y., Liu, B., Dai, W., Cranor, L. F., & Ur, B. Study on User's Attitude and Behavior towards Android Application Update Notification