*software*

MDPI

*Article*

# E-SERS: An Enhanced Approach to Trust-Based Ranking of Apps

**Nahida Chowdhury *, Ayush Maharjan and Rajeev R. Raje**

Department of Computer and Information Science, Indiana University—Purdue University Indianapolis IUPUI, Indianapolis, IN 46202, USA; ius.maharjan@gmail.com (A.M.); rraje@iu.edu (R.R.R.)
* Correspondence: nahida.knu@gmail.com; Tel.: +1-(317)-737-8453

**Abstract:** The number of mobile applications ("Apps") has grown significantly in recent years. App Stores rank/recommend Apps based on factors such as average star ratings and the number of installs. Such rankings do not focus on the internal artifacts of Apps (e.g., security vulnerabilities). If internal artifacts are ignored, users may fail to estimate the potential risks associated with installing Apps. In this research, we present a framework called E-SERS (**E**nhanced **S**ecurity-related and **E**vidence-based **R**anking **S**cheme) for comparing Android Apps that offer similar functionalities. E-SERS uses internal and external artifacts of Apps in the ranking process. E-SERS is a significant enhancement of our past evidence-based ranking framework called SERS. We have evaluated E-SERS on publicly accessible Apps from the Google Play Store and compared our rankings with prevalent ranking techniques. Our experiments demonstrate that E-SERS, leveraging its holistic approach, excels in identifying malicious Apps and consistently outperforms existing alternatives in ranking accuracy. By emphasizing comprehensive assessment, E-SERS empowers users, particularly those less experienced with technology, to make informed decisions and avoid potentially harmful Apps. This contribution addresses a critical gap in current App-ranking methodologies, enhancing the safety and security of today's technologically dependent society.

**Keywords:** trust; subjective logic; sentiment analysis; Google Play Store rankings; static taint analysis; static analysis

check for updates

**Citation:** Chowdhury, N.; Maharjan, A.; Raje, R.R. E-SERS: An Enhanced Approach to Trust-Based Ranking of Apps. *Software* **2024**, *3*, 250–270. https://doi.org/10.3390/software3030013

Academic Editor: Hongyu Zhang

Received: 12 June 2024
Revised: 2 July 2024
Accepted: 10 July 2024
Published: 13 July 2024

**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

Mobile applications ("Apps") markets ("App Stores"), such as the Google Play Store, Apple App Store, Amazon App Store, and Windows Phone App Store, currently have more than 5 million Apps (https://www.statista.com/statistics/276623/number-of-app savailable-in-leading-app-stores/, accessed on 4 March 2021). These markets provide reviews and star ratings of Apps on a scale from 1 to 5 and use the weighted average star ratings score to promote specific Apps [1]. Many studies have indicated that App ratings and associated reviews correlate positively with downloads and sales of Apps [2–6]. To assess this premise, we created a simple survey with one question—"*in general, what is the most important factor that users considered to assess an App before downloading?*"—and distributed the survey to attendees of the IEEE TPS conference in 2019 who attended our session. The participants were randomly chosen for this survey and consisted of academics, students, and practitioners. The survey was conducted anonymously, and we did not request the users to provide their demographic data. We received 130 responses. The response summary is given below.

We recognize that the sample size and the nature of participants in our survey was rather small and homogeneous, but the responses (as indicated in Figure 1) show a similar outcome as that described by Lin et al. [7]. As reviews and rating scores are important to select an App, developers try to manipulate these two factors. Third parties also provide App-promoting assistance (e.g., fake reviews [8]) and guarantee a certain rank desired by the developers for a certain time. In addition, user-provided rating scores have

limitations—the average rating is often influenced by users' two extreme preferences of either one star or five stars [9].

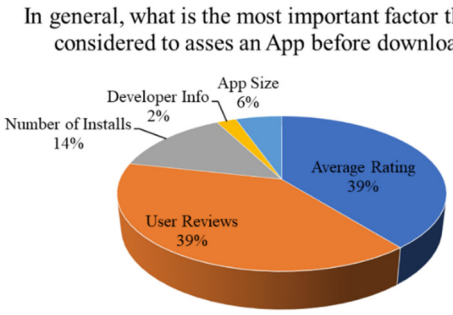



**Figure 1.** Survey response on App evaluating factors.

In our opinion, the average star rating score is not comprehensive enough for selecting a particular App, as the star ratings are not always consistent with the user comments and many times, these comments tend to be unstructured and less focused on the technical aspects of the Apps [10]. In addition, we have found, for many Apps, the average ratings, as evident from the associated narratives [10], do not address issues related to security risks (e.g., data leakage). Many Apps provide personalized services (e.g., SMS services) to the users. Such Apps usually ask users for explicit permissions to obtain personal information (e.g., contact details). A wrong setting of permissions may result in potential risks associated with the unintended disclosure of their sensitive data—malicious Apps have been reported in numerous studies [11–16]. Once a user's data is compromised, they may incur significant hardship while trying to contain the impact of the exposure. In addition, as we had highlighted [10], there tends to be a disparity between the internal (e.g., programmatic features) and external views (e.g., user reviews) of Apps.

The above discussion indicates a need for a comprehensive ranking approach that will encompass several factors, including the trust about the behavior of the Apps. Such an approach will enable users to pick a trustworthy App from the choices. We had proposed an approach, SERS (**S**ecurity-related and **E**vidence-based **R**anking **S**cheme), that addressed this need [17,18]. SERS uses principles of theory of evidence [19], subjective logic (SL) [20,21], static taint analysis, and natural language processing. The trust of an App, in SERS, is defined as the ability of an App to deliver the promised behavior under various operating situations and not to disclose any critical data. SERS computes a comprehensive trust score for an App by considering its internal and external artifacts (we recognize that Apps-related cybersecurity is a vast topic with many facets. Our focus in that study (and this as well) has been rather narrow, related to providing a holistic view that considers internal and external factors of an App and its role in ranking similar Apps. We are of the opinion that such a view, as it considers multiple evidences, will empower the users to make a proper selection out of the available choices for their specific needs).

SERS, however, did not consider the presence of multiple sources to generate evidence, temporal and reputational features of user reviews, and the reputation of the sources used to generate internal evidence. Here, we describe E(**E**nhanced)-SERS, which specifically addresses these three issues. To examine the acceptance of these enhancements, we conducted another informal survey with the same audience and asked the following question: "*Which one of the following ranking schemes could be the right fit to evaluate an App?*". We, again, received 130 responses. These responses indicated that a combined ranking scheme (43.8%) is more acceptable than rankings solely based on average user rating, users' review sentiments, internal factors, and external factors. The principles behind E-SERS are generic, but this study evaluates the principles and the prototype in the Google Play Store context and compares it with other ranking techniques. Future work may extend to applying E-SERS to other App Stores. Hence, the specific contributions of this paper are as follows:

(i) E-SERS formalizes SERS so that it can support any number of sources for generating the necessary evidence for a given App.
(ii) This framework includes a reputation score for each of the sources used to generate internal and external evidence.
(iii) The system features an enhanced risk assessment matrix associated with user permissions.
(iv) The methodology quantifies and uses temporal and reputational aspects of user reviews.
(v) The approach incorporates the feedback from surveys within the computing community, highlighting the preference for combined ranking schemes over simplistic rating-based approaches.

In this study, we address the problem of ranking similar Apps by considering a holistic view and empirically evaluating the proposed approach by using Apps from the Google Play Store. The rest of the paper is organized as follows: Section 2 provides related efforts. Section 3 discusses the E-SERS framework. Section 4 presents the evaluation of E-SERS. Section 5 presents experimental results. Finally, Section 6 states the threats to the validity and concludes the paper by indicating the summary of the research.

## 2. Related Literature

**Sentiment analysis (SA):** SA has been used to analyze reviews about products and movies [22–24]. A few studies have also applied SA to App Store reviews [25,26]. Sangani et al. [27] applied the review-to-topic mapping approach to pinpoint the most demanded feature, by the users, of Apps. Pagano and Maalej [3] and Palomba et al. [28] have examined the types of user feedback and unveiled how developers monitor user reviews and correlate them to users' ratings. A few research efforts have computed trust tuples based on the reviews of Apps [29,30]. These efforts have focused only on the user reviews—we, in E-SERS, combine internal and external views of Apps to generate trust quantification of Apps.

**Data flow analysis:** User permissions play an important role [31] in identifying possible malicious activities of Apps. There are studies (e.g., DroidRanger [32] and DroidRisk [33]) that have assessed permission-based risks of Apps. DroidRisk considers the frequency and the number of permissions an App requires. Sarma et al. [31] and Gates et al. [34] have assigned high-risk quantification to severe permissions. However, permissions alone are not sufficient to assess and quantify risks, as not all requested permissions are actively utilized during the execution [35]. In E-SERS, we focus only on the faulty data flows and corresponding permissions—like the approach suggested by Mirzaei et al. [36] to categorize the data flows into benign and malicious classes.

Static taint analysis locates sensitive APIs—a few prominent static taint analysis tools are FlowDroid [37], TaintDroid [38], AndroidLeaks [39], and DroidSafe [40]. FlowDroid performs better than other tools while identifying data leaks. Hence, E-SERS uses FlowDroid to collect Direct Trust Artifacts (DTAs).

**Trust:** Trust has been studied in networks [41], the Internet of Things [42], and social [43] and legal [44] communities—trust is established between the trustor and trustee through observing prior events [20,45]. In our past work [46], we had presented a comprehensive survey of trust in the software domain. In [30], we developed a trust model that is based on subjective logic for incorporating trust with events [47]. Here, we have enhanced our previous models [10,17,18,30] and formalized the evidence-based trust management framework to infer direct and indirect trust artifacts for any given App.

**Fraud act detection:** Hernandez et al. [48] presented the 'Racketstore' platform, which collects App usage details and reviews them to detect any fraudulent activity that an App's developer may practice to increase the rank of their App. Here, the authors' approach is completely based on the indirect trust artifacts of an App. In [49], the authors have proposed a methodology to increase the trustworthiness of user engagement metrics (e.g., number of installs) by identifying the incentivized app's installation which is also based on external artifacts (e.g., offer details). E-SERS focuses on both direct and indirect trust

artifacts. It aims to empower users by providing the trust score of an App instead of reporting fraud acts or discovering App store policy violations.

**Traditional methods for App ratings:** Popular App Stores, such as Google Play Store, provide an average star rating (between 1 and 5) for an App based on individual user ratings. E-SERS uses a more comprehensive scheme to rate Apps.

**Ranking of Apps:** Existing research efforts (e.g., [50,51]) are either based on an internal view or external view. Zhu et al. [50] presented a hybrid ranking principle which was a combination of risk scores and overall rating. The risk factor is established based on the permission requested by the App and risk value is determined by examining each of the dangerous permissions Apps request. Using permissions alone to estimate risk has serious limitations and is inaccurate. Cen et al. [51] used a crowd-sourcing ranking approach to solve the App risk assessment problem from users' comments. However, users' comments are subjective—thus, E-SERS focuses on both programmatic and user perspectives of an App.

## 3. E-SERS Design

### *3.1. Architecture*

The conceptual architecture of the E-SERS is illustrated in Figure 2 (discussed in this section). The four basic components of E-SERS, and the notations that we use throughout the paper, are as follows:

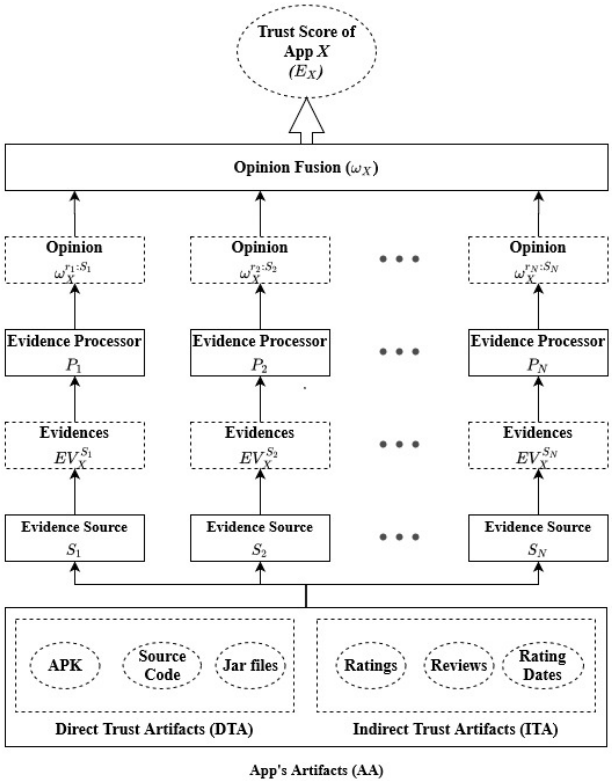



**Figure 2.** E-SERS Architecture.

**App's Artifacts (AAs):** The AAs are categorized into "Direct Trust Artifacts" (DTAs) and "Indirect Trust Artifacts" (ITAs). DTAs indicate various internal evidence about an App and are gathered from APK files, source code, and jar files of an App. In contrast, user opinions, such as ratings and reviews, contribute to the ITAs of an App.

**Evidence Sources:** The evidence source set, S, for an App X is divided into two mutually exclusive subsets, $S_{DT}$ and $S_{IT}$, which denote the list of sources that are used to generate the DTAs and the ITAs, respectively. Each evidence source ($S_i \in S$) generates a set

of evidence, $EV_X{}^{Si} = \{ev_1, \ldots, ev_n\}$. Each evidence, $ev_i$, can be positive, negative, or neutral. Different techniques are used for extracting various types of evidence.

**Evidence Processors:** Each $S_i$, for an App X, has an associated evidence processor, $EP_i$. An $EP_i$ maps the set of evidence, $EV_X{}^{Si}$, to an opinion $\omega_X{}^S i$. Each source may produce different evidence and, therefore, before fusing such different opinions, we need to normalize them so that evidence from a reputed source has more weight than from non-reputed ones. To do so, we have introduced the concept of source reputation into E-SERS. The reputation of each source, $\omega_{Si}{}^r{}_i$, is combined with the opinion of $\omega_X{}^S i$ to compute the weighted opinion $\omega^r{}_i{}^{:S}{}_i$. Like the technique suggested in [21], we use the discounting (or weighted) operator ($\otimes$) to represent the degree of trust about an evidence source.

**Opinion Fusion:** Opinions from different sources, $\omega_X{}^S{}_1, \ldots, \omega_X{}^S{}_n$, can be combined into a single opinion ($\omega_X{}^S$), using the consensus operator ($\oplus$) [52]. However, the consensus operator treats opinions equally—hence, in E-SERS, we have used the cumulative weighted fusion operator [53] to combine opinions and create a trust score for an App.

### *3.2. Evidence-Based Trust Rating Algorithm*

#### 3.2.1. Evidence to Opinion Mapping

We use SL to represent an opinion about an App. The opinion about an App X, created by a source Si, ($\omega_X{}^S i$) is indicated by a ($b$, $d$, $u$, $a$) tuple. Here, $b$, $d$, and $u$ represent the belief, disbelief, and uncertainty that a proposition—that we can trust the App X—is true, and a is the base rate that the proposition is correct, in the absence of any evidence. The ($b$, $d$, $u$, $a$) tuple is calculated using the following equations [54]:

$$b = \frac{postive\ evidence}{total\ evidence + n} \tag{1}$$

$$d = \frac{negative\ evidence}{total\ evidence + n} \tag{2}$$

$$u = \frac{n}{total\ evidence + n} \tag{3}$$

$$a = \frac{1}{n} \tag{4}$$

In these formulae, '$n$' indicates possible outcomes about any evidence. In E-SERS, '$n$' is equal to 2 because an evidence can either be present or absent in an App. A trust score of an App X, obtained from an opinion, $\omega_X$, is measured as the expected value ($E_X$) that indicates the probability that X is trustworthy and is calculated as:

$$Ex = b + a * u \tag{5}$$

#### 3.2.2. Algorithms for Computing the Trust Score for an App X

Algorithm 1 accepts DTAs, ITAs, sources of evidence, and the user-desired weights ($\alpha$ and $\beta$) for both views of an App X and computes its trust score.

**Algorithm 1. Computation of the Trust Score for an App**

***procedure calculateTrustScore*** ($DTA_X$, $ITA_X$, $S_{DT}$, $S_{IT}$, $\alpha$, $\beta$)
#Generate internal opinion from DTA for an App X.
$\omega_X{}^{\oplus S}{}_{DT} \leftarrow$ *create_internal_opinion* ($DTA_X$, $S_{DT}$).
#Generate external opinion from ITA for an App X.
$\omega_X{}^{\oplus S}{}_{IT} \leftarrow$ *create_external_opinion* ($ITA_X$, $S_{IT}$).
#Apply weighted consensus operator
$\omega_X{}^{\oplus\,(S}{}_{DT}, SIT) \leftarrow$ *weighted_fusion* ($\omega_X{}^{\oplus S}{}_{DT}$, $\omega_X{}^{\oplus S}{}_{IT}$, $\alpha$, $\beta$).
#Apply Formula (5) to compute expected value and normalize
$E_X \leftarrow E$ ($\omega_X{}^{\oplus\,(S}{}_{DT'}{}^{SIT)}$).
*return normalized* $||E_X||^2$ *to scale 5.*

Algorithm 2 maps input $DTA_X$ to the direct trust-based tuple, ${\omega_X}^{\oplus S}{}_{DT}$. Different evidence, generated from $S_{DT}$, is classified as positive or negative based on their behavior towards the App. Equations (1) to (4) are then used to compute the direct trust tuple of the App. The discounting operator is used to combine a source's reputation with ${\omega_X}^S i$ to compute ${\omega^r}_i{}^{:Si}$. Opinions from all sources are merged using the consensus operator to create a single opinion, ${\omega_X}^{\oplus S}{}_{DT}$. Algorithm 3 maps input $ITA_X$ to the indirect trust tuple, ${\omega_X}^{\oplus S}{}_{IT}$. Here, for each evidence, the reputation of each review and the associated temporal weight are used to determine the influence of the evidence on the indirect trust tuple of the App.

**Algorithm 2. Computation of opinion from DTA**

***procedure createInternalOpinion*** ($DTA_X$, $S_{DT}$)
  *for* $S_i \in S_{DT}$ *do*
    *positive_evidence* ← *null*
    *negative_evidence* ← *null*
    $S_i$: *ev* (*X*) ← *generate_internal_evidence*(*X*)
    *for e* ∈ $S_i$: *ev*(*X*)! = *null do*
      *if e is positive evidence*
      *positive_evidence++.*
    *else*
      *negative_evidence++.*
    *end for*
    *Apply Formulae (1) to (4) to determine (b, d, u, a),* ${\omega_X}^{Si}$.
    *Evaluate reputation* ($r_i$) *of* $S_i$ *based on F1-Score,* ${\omega_{Si}}^{ri}$.
    *Calculate weighted opinion of* $S_i$, ${\omega_X}^{ri:\ Si,}$ *using the discounting operator.*
  *end for*
    *Apply consensus operator to fuse opinions from different sources and compute* ${\omega_X}^{\oplus S}{}_{DT}$.

**Algorithm 3. Computation of opinion from ITA**

*procedure createExternalOpinion* ($ITA_X$, $S_{IT}$)
  *for* $S_i \in S_{IT}$ *do*
    *positive_evidence* ← *null.*
    *negative_evidence* ← *null.*
    $S_i$: *ev* (*X*) ← *generate_external_evidence*(*X*).
    *for e* ∈ $S_i$: *ev*(*X*)! = *null do*
      *review_reputation_weight* ← *apply Formula (6)*
      *# Normalized to scale 10*
  ||*temporal weight*||$^2$ ← *assign highest score to to recent reviews*
      *weight[e]* ← *review_reputation_weight* ×*temporal_weight.*
      *if (e is positive_evidence)*
        *positive_evidence* += *weight[e]*
      *else*
        *negative_evidence* += *weight[e]*
      *end for*
      *Apply formulae (1) to (4) to determine (b, d, u, a),* ${\omega_X}^{Si}$.
    *Evaluate reputation* ($r_i$) *of* $S_i$ *based on F1-Score,* ${\omega_{Si}}^{ri}$.
      *Calculate weighted opinion of* $S_i$, ${\omega_X}^{ri:\ Si,}$ *using the discounting operator.*
    *end for*
    *Apply consensus operator to fuse opinions from different sources and compute* ${\omega_X}^{\oplus S}{}_{IT}$.

## 4. E-SERS Approach and Evaluation

We created a prototype, using the above algorithms, and empirically evaluated it in the context of the Google Play Store. We identified five popular categories in the Google Play Store—Shopping, Travel, Insurance, Finance, and News. From these categories, we selected 25 Apps for this empirical evaluation.

### *4.1. Computation of Direct Trust*

To generate the DTAs of an App from the above set of Apps, like [18], we used an open source static taint analyzer tool, FlowDroid [37]—other researchers have also used this tool [36,55]. FlowDroid traces sensitive information associated with an App by identifying source–sink pairs. It then returns detailed information (e.g., API method's name that tries to read/write sensitive information from the App to third parties) about unauthorized leaks of any confidential data. In our experimentation, we also used another tool (FindBugs). For the sake of brevity, here we only discuss the results obtained from FlowDroid. Any identified leaks are considered as internal evidence. This set of evidence is expressed as ${\omega_X}^S{}_1$ and is mapped to the internal trust tuple as described below.

#### 4.1.1. Evidence Mapping to Trust Tuple Creation

**Mapping Evidence of $S_1$ to ${\omega_X}^S{}_1$.** In [18], we had introduced a four-step analysis for mapping sensitive data leaks to trust tuples. It consisted of: (i) identifying sensitive source–sink pairs using FlowDroid, (ii) classifying sources and sinks into various categories using SuSi [56], (iii) assessing the risk factors with these pairs using NIST guidelines [57,58], and (iv) computing the internal trust tuple using Equations (1)–(4). The step (iii) is enhanced in E-SERS by employing a $4 \times 4$ risk assessment matrix (as opposed to the preliminary $3 \times 3$ heuristic-based matrix used in [17,18]) and is used in the step (iv). Hence, below we describe only steps (iii) and (iv).

**Step (iii):** In this step, we assess the risk factors associated with permissions that are given to sensitive APIs. Android divides the permissions into different protection levels that affect whether run-time permission requests are required or not. Potential risks using the permissions are characterized as *Normal*, *Signature*, and *Dangerous*. We collected 91 permission identifiers (36 Normal permission identifiers, 29 Signature permission identifiers, and 26 Dangerous permission identifiers) from the Android site [59] and mapped them to the corresponding APIs using PScout [60], which is a technique to map API calls to permissions identifiers.

NIST guidelines for risk management of information technology systems ([56,57]) are followed to assess the quantitative risk associated with the Android permissions. According to these guidelines, risk assessment is defined as:

$$R(P) = L(P) \times I(P) \tag{6}$$

where $P$ is the requested permission, $R(P)$ is the risk of $P$, $L(P)$, and $I(P)$ are the likelihood, and the impact of $P$. **Likelihood** indicates the probability that a potential vulnerability may be exercised within the construct of the threat environment. **Impact** is used to measure the level of risk resulting from a successful threat exercise of a vulnerability. The determination of these risk levels is subjective, due to the assignment of a probability to the likelihood of each threat level and a value for its impact. We used the enhanced risk assessment matrix shown in Table 1—it contains four levels of likelihood and impact.

**Table 1.** Quantitative $4 \times 4$ risk assessment matrix.

| Likelihood (Source/Sink) | Catastrophic Impact (100) | Critical Impact (50) | Marginal Impact (20) | Negligible Impact (10) |
|---|---|---|---|---|
| Frequent (1.0) | High (100) | Moderate (50) | Moderate (20) | Low (10) |
| Probable (0.5) | Moderate (50) | Moderate (50) | Low (10) | Low (5) |
| Remote (0.2) | Moderate (20) | Low (10) | Low (4) | Low (2) |
| Improbable (0.1) | Low (10) | Low (5) | Low (2) | Low (1) |

By applying Equation (6), the risk assessment scale is divided into three distinct categories: *High (>50 to 100)*, *Moderate (>10 to 50)*, and *Low (1 to 10)*. Based on the permission

that is requested, the level of impact is classified into four different categories: *Catastrophic* (identifiers that fall into the Dangerous permission identifiers category), *Critical* (identifiers that fall into the Significant permission identifiers category), *Marginal* (identifiers that fall into the Normal permission identifiers category), and *Negligible* (identifiers that do not belong to any of the permission identifiers categories). The source and sink categories are placed into different likelihood categories based on their appearance in the App's source code.

We selected three malware data sets, one from VirusShare (https://virusshare.com) and two others from Drebin (https://drebin.mlsec.org/), which contain a total of 2555 malicious Apps, accessed on 6 March 2021. On these Apps, we ran FlowDroid and stored the source and sink categories that it reported. If any of the source/sink categories appear in these three malware data sets, then those are classified as belonging to the Frequent class; if they appear in two of the observed data sets, then those belong to the Probable class; if they appear in only one data set, it is classified into the Remote class; and the rest of the categories are considered as belonging to the Improbable class. The assignment of source and sink distribution to different likelihood categories is given below in Table 2.

**Table 2.** Likelihood categorization based on appearance.

| Likelihood | Source Category | Sink Category |
|---|---|---|
| Frequent (1.0) | ACCOUNT_INFORMATION<br>LOCATION_INFORMATION<br>NETWORK_INFORMATION<br>NO_CATEGORIES<br>UNIQUE_INFORMATION | LOG<br>NETWORK<br>NO_CATEGORIES<br>SMS_MMS |
| Probable (0.5) | DATABASE_INFORMATION<br>FILE_INFORMATION | ACCOUNT_SETTINGS<br>FILE<br>CONTACT_INFORMATION |
| Remote (0.2) | CONTACT_INFORMATION<br>NFC<br>UNIQUE_INFORMATION | CALENDAR_INFORMATION<br>SYSTEM_SETTINGS |
| Improbable (0.1) | Rest of the Source categories | Rest of the Sink categories |

**Step (iv)**: Any evidence that ensures data confidentiality is positive evidence and one that involves information leakage is negative evidence. Along with the analysis report generated by FlowDroid, we also keep track of the run-time log file. From that log file, we extract the number of total sources (*ST*) that exists in an App's code. If there is no leak, then *ST* is considered as a total number of positive evidence. Again, if data leaks are found then the positive evidence is calculated by subtracting the number of faulty sources (*SF*) from *ST*, where *SF* indicates the sources involved in information leakage. Once all evidence is generated, then Equations (1)–(4) are applied to compute the $(b, d, u, a)$ tuple that reflects the opinion ${\omega_X}^S{}_2$.

#### 4.1.2. Computing Opinion of Direct Trust

After computing the opinion of FlowDroid (henceforth referred as $S_1$), $\omega_X^{S_1}$, about an App X, we then evaluate the reputation of $S_{1,}$ indicated as $\omega_{S_1}^{r_1}$. We have used precision, recall and F-Score to compute $\omega_{S_1}^{r_1}$. To assess FlowDroid (i.e., $S_1$), DroidBench [61] is utilized [36]. The result of this benchmarking effort is presented in Table 3 [37]. The reputation tuple is based on the F1-score, as it gives a better measure of the wrongly classified instances than the accuracy metric [62]. The F1-score is considered as the value of belief and the rest is assigned to disbelief. Here, the uncertainty remains zero due to the assumption that the domain experts formulate the benchmarks, so there is hardly any chance for ambiguity. The reputation scores of S1, using the details provided in Table 3, are (0.89, 0.11, 0, 0.5). Next, Equation (5) is applied to compute the $\omega_{S_1}^{r_1}$. To compute the direct

trust of Apps, we used a single source (FlowDroid) to generate evidence; hence, the fusion of opinions is not required here ($\omega_{S_1}^{r_1} \Leftrightarrow \omega_X^{\oplus S_{DT}}$).

**Table 3.** Reputation of SDT.

| Source ($S_{DT} \subset S$) | Precision ($p$) | Recall ($r$) | F1-Score ($2pr/(p + r)$) | Reputation ($b$, $d$, $u$, $a$) |
|---|---|---|---|---|
| FlowDroid ($S_1$) | 0.86 | 0.93 | 0.89 | (0.89, 0.11, 0, 0.5) |

### *4.2. Computation of Indirect Trust*

#### 4.2.1. Data Collection and Pre-Processing

We selected five Apps, each from the Shopping, Travel, Insurance, Finance, and News categories in the Google Play Store. These categories have been identified by NowSecure (https://www.nowsecure.com/), a leading security company, in their research efforts [10,63], accessed on 7 March 2021. There are other solutions that detect harmful viruses present in Apps (e.g., Google Play Protect— https://developers.google.com/android/play-protect, accessed on 7 March 2021). However, the warnings generated by such alternatives are not quantifiable. NowSecure generates a risk score for an App—this score is based on the Common Vulnerability Scoring System (CVSS). The major difference between our approach and NewSecure is that we have introduced a mapping scheme to compute a CVSS score for the good practices too. We investigated the association between NowSecure, and the insights based on DTAs—we do not have a subscription to NowSecure's paid service, so we could not gather any evidence about the Apps in the data set. In each of these five categories, we selected one App that was used by NowSecure in their study. After that, we identified four other Apps that were "similar in functionality" (indicated by the Google Play Store) to that App and had a reasonable number (average number of reviews per App is 2100) of user reviews. The data set that we created for experimentation contained reviews from 23 July to 19 October 2019. For each App, we collected three different data items using an in-house review crawler: the App's basic details (e.g., user rating, total number of reviews and installs, etc.), its *Newest reviews*, and the *Most Relevant reviews*. Google Play Store characterizes an App's reviews into three distinct categories: *Newest*, *Most Relevant*, and *Rating*—we focused only on the *Newest* and *Most Relevant* data sets. Reviews are converted to Unicode and then stored. Before sentiment analysis, reviews are decoded, via the Unicode data library [64], to remove umlauts, accents, and other similar features.

#### 4.2.2. Mapping Sentiment Value to Opinion Model

The IBM Watson natural language understanding [65] tool ("Watson" is also denoted as $S_2$ in the following discussion) is used to predict the sentiment of preprocessed reviews. Watson returns a sentiment score in the range of $[-1, +1]$ and indicates whether a given review reflects the positive or negative sentiment of the user. Watson's opinion is mapped to compute $\omega_X{}^{S}{}_2 = (b_X{}^{S}{}_2; d_X{}^{S}{}_2; u_X{}^{S}{}_2; a_X{}^{S}{}_2)$ as discussed below.

#### 4.2.3. Sentiment Score to (b, d, u) Tuple Mapping

We followed a conversion scheme with boundary cases, like Gallege [29], while mapping Watson's opinion to $\omega_X{}^{S}{}_2$. However, they used the linear regression model, whereas we used the random forest regression model [66], as the mean absolute error is typically higher for linear regression than the random forest regression model. Table 4 contains the boundary cases for converting textual sentiments—from a two-tuple of sentiment score to (b, d, u). Here, (0, 1, 0) represents extreme disbelief and (1, 0, 0) represents extreme belief about a review. These boundary cases are fed into a random forest regression model to predict b and d, then compute u.

**Table 4.** Mapping of sentiment score to (b, d, u).

| Sentiment Score | (b, d, u) | Sentiment Score | (b, d, u) |
|---|---|---|---|
| −1 | (0, 1, 0) | +1 | (1, 0, 0) |
| −0.75 | (0, 0.75, 0.25) | +0.75 | (0.75, 0, 0.25) |
| −0.5 | (0, 0.5, 0.5) | +0.5 | (0.5, 0, 0.5) |
| −0.25 | (0, 0.25, 0.75) | +0.25 | (0.25, 0, 0.75) |

#### 4.2.4. Review Reputation

To determine the reputation of reviews, researchers have adopted reviewer-centric methods [67,68]. Such a reviewer-centric approach is not feasible, as Google Play Store does not provide reviewer details. Hence, we used a review-centric approach to determine the reputation of reviews. The Most Relevant category contains the set of reviews that were *liked* by the other users. We use this category to establish the reputation of any review—we utilize the '*num of likes*' and the '*sentiment score*' of the Most Relevant reviews. Next, the mapping mechanism mentioned above is applied to convert the sentiment score of a review to a (b, d, u) tuple. The (b, d, u) tuples of Most relevant reviews are clustered (using k-means [69]) into different clusters ($C_1; C_2; \ldots; CN$). Finally, the average number of '*total likes*' ($L$) for all reviews ($\forall$r) that belong to a cluster Ci is used as a weight for that cluster and computed as

$$W_{C_i} = \frac{\sum_{\forall r \in C_i} L_r}{\sum_{\forall r \in C_i} r} \tag{7}$$

Once the weight is determined for each cluster, we predict the cluster membership for reviews in the *Newest* data set. Based on the cluster determination, the corresponding weight is assigned to each review. A high value of the weight represents a highly reputed review, and a low value denotes lower importance of that review (probably a fake review). Thus, this approach reduces the influence of fake reviews while computing the trust score of an App.

#### 4.2.5. Determination of Temporal Weight

App developers routinely release new versions which fix bugs and update features. Therefore, it is appropriate to treat the old and recent reviews differently. We have introduced a temporal weight for each review to reduce the impact of older reviews. The weight is determined by Hawkes processes; a self-exciting spatio-temporal point processes model [70]. To this model, we feed the timestamps of reviews from the *Newest* reviews data set. Then, the model learns to exponentially weigh reviews, going back in time, and returns the corresponding weight for each timestamp. We have, for simplicity, normalized the temporal weights to a scale of 10.

#### 4.2.6. Computing Opinion of Indirect Trust

Three elements are required to determine $\omega_X^{S_2}$: the review sentiment score, the temporal weight, and the weight of the review reputation. By multiplying these two weights, we compute the total weight for a review [71]. A review that has a sentiment score between 0 and 1 is considered as positive evidence, and between 0 and −1 as negative evidence. After generating all evidence, (1)–(4) are applied to compute the (b, d, u, a) tuple that indicates the opinion $\omega_X^{S_2}$.

After computing the opinion of the tool $S_2$, we need to evaluate its reputation. The existing literature provides Watson's (i.e., $S_2$) F1-score for data sets, such as movie reviews and Twitter comments. As reviews of Apps are conceptually different than these data sets, to assess $S_2$, we created a benchmark based on collected reviews. We asked four domain experts to manually label the sentiment of 750 reviews—a total of 3000 reviews. To ensure the quality of the labels, we exchanged the reviews between these experts and cross-verified the outcomes. If a discrepancy was observed, then, based on the majority

judgment, the review was labeled accordingly. From this labeled data set, we randomly picked 1000 positive reviews and 1000 negative reviews to create the benchmarking set—the confusion matrix for this benchmark data set is shown in Table 5. Based on this matrix, the *Precision* and *Recall* values for Watson are 0.89 and 0.85 and the *F1*-score of Watson is 0.87. Thus, the reputation $S_2$ ($\omega_{S_2}^{r_2}$) is (0.87, 0.13, 0, 0.5). Next, the discounting operator is applied to compute the $\omega_X^{r_2:S_2}$. To compute the opinion of indirect trust, we have used a single source (Watson) to generate evidence; hence, the fusion of opinions is not required here ($\omega_X^{r_2:S_2} \leftrightarrow \omega_X^{\oplus S_{IT}}$).

**Table 5.** Watson confusion matrix.

| | **Positive (Actual)** | **Negative (Actual)** |
|---|---|---|
| Positive (Predicted) | 853 (TP) | 99 (FP) |
| Negative (Predicted) | 147 (FN) | 901 (TN) |

### *4.3. Evidence Processor and Opinion Fusion*

After computing the opinions for the direct trust ($\omega_X^{\oplus S_{DT}}$) and indirect trust ($\omega_X^{\oplus S_{IT}}$) of an App, we combine them into a single opinion, using the cumulative weighted fusion operator. The direct trust-based evidence is likely to have less ambiguity, as it solely focuses on the functional perspectives of an App. Hence, we assign a lower weight to the $\omega_X^{\oplus S_{IT}}$ than to the $\omega_X^{\oplus S_{DT}}$; the assigned weights are 30% and 70%, respectively. These weights can be adjusted as the user desires (wee understand that a user's ability to tolerate risk, and hence the trust in an unknown App, is subjective and also depends upon their technical background. Thus, the notion of trust is inherently user-dependent. What E-SERS provides, to users, is a framework that considers many facets of any App. The trust scores, and hence the rankings, of similar Apps provided by E-SERS is intended to empower users in their selection process—the user is given a choice of weighting internal and external evidence as per their preference and that will affect the ranking of similar Apps). This resultant opinion ($\omega_X^{\oplus(S_{DT},\, S_{IT})}$) counts all available evidence and thus, provides a more reliable quantification of trust associated with each App than the average star ratings provided by the Google Play Store. The $\omega_X^{\oplus(S_{DT},\, S_{IT})}$ allows us to calculate the trust score ($E_X$) using Equation (5), which is normalized to a scale of 5. The value of $E_X$ helps to rank-order similar Apps. The ranking generated by E-SERS is compared using the Kendall Tau distance method [72] with other alternatives.

## 5. Experimental Results

In our study, we have created a data set of 25 popular Android Apps from distinct categories that are available in the Google Play Store. We selected the categories of Apps that have been identified by NowSecure in their research effort [10,63]. In our study, Apps are selected from the Shopping, Travel, Insurance, Finance, and National and Local News categories. From each category, five different Apps were picked for our experiments. In each category, we selected one App that has been used by NowSecure in their study. After that, we identified four other Apps that were "similar in functionality" (as indicated by the Google Play Store) to that App and had a reasonable number (average number of reviews per App is 2100) of user reviews. These selected Apps belong to different ranges of popularity (such as the most popular, popular, and less popular) in terms of the number of installs. In our previous work [18], we had addressed the correlation between the traditional star rating, popularity (number of installs), and trust of an App. In [18], we performed our experiment by applying a data set of 35 Apps, taken from Google Play Store. The data set [18] indicated the following behavior:

- If we consider only the traditional star ratings of all the Apps, as a typical App user would, we find that there is hardly any difference between Apps; however, the number

of installs for each App varies a lot. This highlights the fact that traditional star rating does not accurately reflect the trust of an App.

- In our experimental data set and based on the associated evidence that SERS generated, a less popular App (in terms of the number of downloads) is assessed as a more secure App than the other popular Apps. So, the SERS will provide users with a comprehensive view of an App and help them to select a more secure App instead of just following the traditional ratings and making a choice.

In the following discussion, we do not disclose Apps' identifiable details to maintain anonymity.

### *5.1. Findings from DTA Sources*

The number of data leaks identified by FlowDroid for each category of Apps along with the reported sources and sinks categories are presented in Table 6. Source and sink APIs that belong to NO CATEGORY are not reported here, as they refer to non-sensitive data flows in SuSi [56]. In [33], authors identify that sources that are categorized into NETWORK INFORMATION and UNIQUE IDENTIFIER are more likely to occur in malware Apps than in benign Apps. In addition, that study indicates that malware Apps are more prone to use short message service (SMS) as a sink to leak data to third parties—such scenarios are found in our test data set too. For the News category Apps, we noticed that the quantities of the source API belonging to the UNIQUE IDENTIFIER category and the sink APIs that refer to SMS_MMS were higher than the other categories. An interesting insight from the direct trust-based result is that the Apps selected from the News category were *more probable to leak sensitive information than the ones from the other categories*. A similar observation had been reported by NowSecure where they indicated that almost all local news Apps (in their data set) leaked user data, whereas 40% of them had severe security vulnerabilities that could lead to sensitive information being compromised.

**Table 6.** Data leaks details generated by FlowDroid.

| App Category | # of Data Leaks | Source Categories | Sink Categories |
|---|---|---|---|
| Shopping | 664 | LOG (239)<br>SMS_MMS (186)<br>NETWORK_INFORMATION (17)<br>FILE (6)<br>LOCATION_INFORMATION (2) | SMS_MMS (93)<br>NETWORK (24)<br>FILE (5)<br>CALENDAR_INFORMATION (4)<br>CONTACT_INFORMATION (3) |
| Travel | 881 | SMS_MMS (68)<br>LOG (63)<br>FILE (8)<br>NETWORK_INFORMATION (3)<br>CALENDAR_INFORMATION (2)<br>ACCOUNT_SETTINGS (1) | SMS_MMS (46)<br>FILE (10)<br>CALENDAR_INFORMATION (2)<br>ACCOUNT_SETTINGS (1) NETWORK (1) |
| Insurance | 635 | SMS_MMS (186)<br>LOG (155)<br>FILE (9)<br>ACCOUNT_SETTINGS (5)<br>NETWORK_INFORMATION (4)<br>CALENDAR_INFORMATION (2) | SMS_MMS (73)<br>NETWORK (16)<br>ACCOUNT_SETTINGS (3)<br>CALENDAR_INFORMATION (3)<br>FILE (2) |
| Finance | 1237 | LOG(l61)<br>SMS_MMS (63)<br>NETWORK_INFORMATION (13)<br>FILE (2) | SMS_MMS (86)<br>NETWORK (9)<br>CALENDAR_INFORMATION (5)<br>LOG (2) |
| News | 1399 | LOG (114)<br>SMS_MMS (80)<br>UNIQUE_IDENTIFIER (14)<br>FILE (9)<br>NETWORK_INFORMATION (8)<br>ACCOUNT_SETTINGS (3) | SMS_MMS (157)<br>NETWORK (18)<br>LOG (13)<br>FILE (6)<br>ACCOUNT_SETTINGS (4)<br>CALENDAR_INFORMATION (3)<br>CONTACT_INFORMATION (1) |

### 5.2. Findings from ITA Sources

As indicated, we have collected a data set of 25 Apps from five distinct categories. The data set of the associated user reviews is described in Table 7. The matrix of average words (per review) denotes that the *Most relevant* reviews are always more detailed than the reviews that are in the *Newest* category.

**Table 7.** Statistics of collected user review data set.

| Newest Review Data Set | | Most Helpful Review Data Set | |
|---|---|---|---|
| Total number of crawled reviews | 52,519 | Total number of crawled reviews | 24,299 |
| Average number of reviews per App | 2100 | Average number of reviews per App | 970 |
| Average words per review | 14.8 | Average words per review | 22.3 |

Figure 3 presents the sentiment scores for each review in our data set where every point denotes the score for an individual review. The box plot shows the median, first, and third quartiles, and minimum and maximum sentiment scores for individual rating on a scale of 1 to 5. However, a significant amount of outliers are evident for the ratings of 1, 2, and 5.

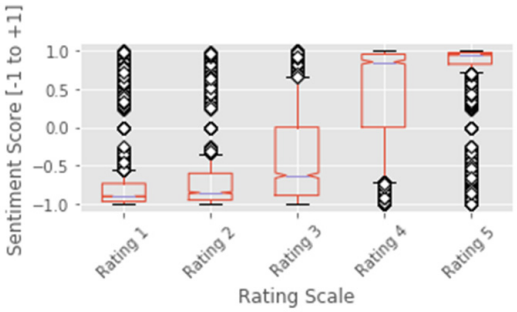



**Figure 3.** User−given rating score vs. review's sentiment score.

After examining the review sentiments and the corresponding ratings, we found mismatches. Consider this review, for example: "*Don't care for this app. Too confusing, even when it works.*". The user provided a rating of 5 for this review, where Watson returned a negative sentiment, reflecting a mismatch. We also performed a review-based evidence analysis between the *Newest* and *Most relevant* reviews data sets, presented in Figure 4. For every category, there is a clear mismatch of evaluation based on these two data sets.

For example, *Newest* reviews of *App2* in the Shopping category mostly indicate positive sentiment whereas feedback in the Most Relevant data set indicates a mix of positive and negative sentiments. However, we have noticed a significant difference in the News category. Here, the sentiment score for each App's *Newest* reviews data set deviates from a high to low sentiment score for the Most Relevant reviews data set. For example, the sentiment score of *App3* in the News category deviates from [0.75, −0.25] to [0, −0.25]. This indicates that in the News category, users are experiencing similar difficulties (such as ads, malware, bugs, etc.) that were previously highlighted by others. Consider the following partial review with a high number of likes in the News Category: "*Used to be 5 stars until ads started popping up. There are ads running continuously on the top of the screen. . .. I have to delete this App because its ruined now. . . ..*"; the number of likes for this comment is **1765** and the sentiment score is **−0.909597**.

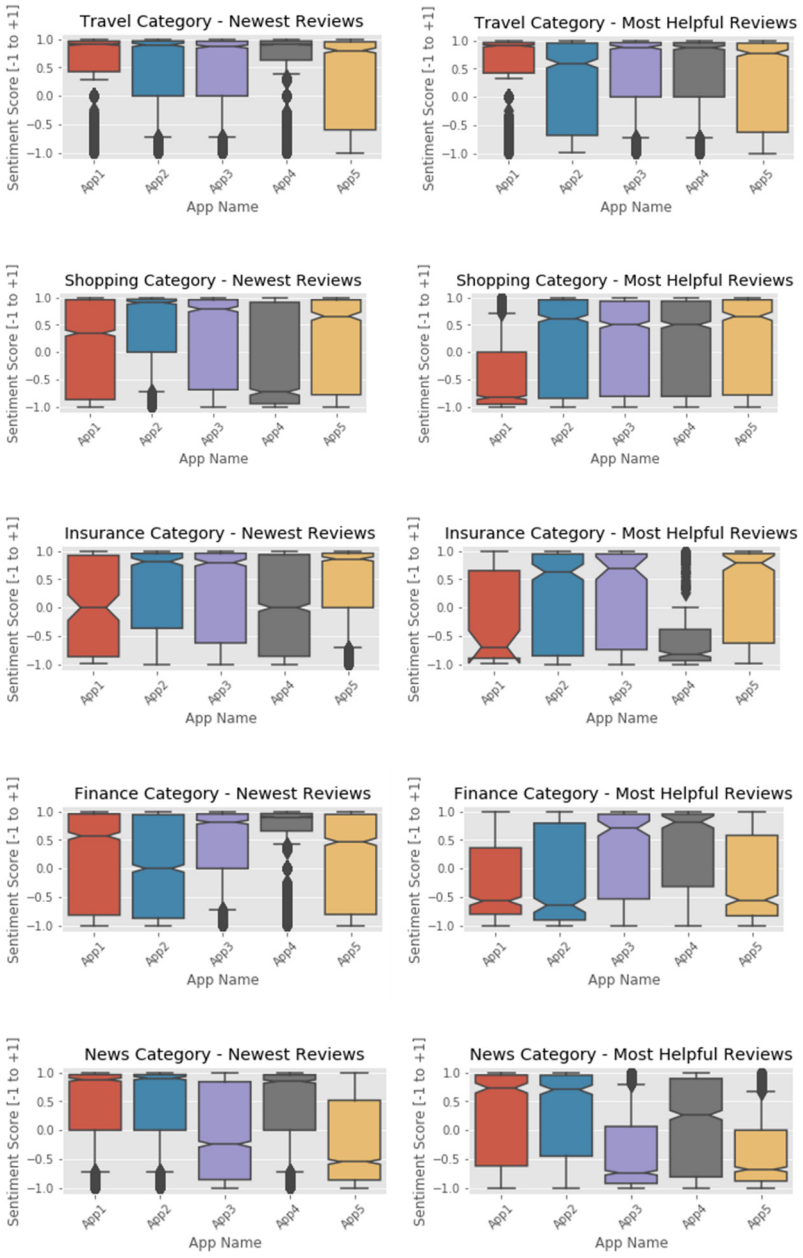



**Figure 4.** Review−based evidence analysis.

From the above discussion, it is inferred that looking only at the *Newest* review data set is not an ideal option, as it fails to unfold the detailed behavior of an App from the users' point of view. Therefore, the user should observe the *Most Relevant* reviews as well. However, in most of the cases in the *Most Relevant* review data set, the sentiment score was found to be negative. We can, hence, for our data set, conclude that the reviews in the *Most Relevant* category tend to have more negative sentiment than the *Newest* category, which reflects that the users are more inclined to like criticism rather than appreciation of an App. Overall, users give *'like's* or write reviews to present their dissatisfaction or problems that they are facing. We also examined the number of reviews specific to bug or security concerns (presented in Table 7). To determine that, we created a list of keywords, which contained *bug, fix, problem, issue, defect, crash, solve, permission, privacy, security, spy, spam, malicious, and leaks*—most of these keywords are described by Maalej et al. [73] under the bug reports review type.

The keyword distribution, shown in Table 8, indicates that the users have addressed more bug-related feedback than security-related concerns. However, the total number of bug- and security-related reviews indicate that typical users are not aware of these internal issues. We found that one of the most popular Apps (*App2*), with more than 10 million installs, in the Finance category actively leaks sensitive user information.

**Table 8.** Reviews related to bug and security scope.

| | | | ***Newest Reviews*** | | |
|---|---|---|---|---|---|
| | Shopping | Travel | Insurance | Finance | News |
| Bug (%) | 9.7 | 11.2 | 9.6 | 7.3 | 8.4 |
| Fix (%) | 33 | 24.7 | 33.9 | 34 | 43.2 |
| Problem (%) | 23.1 | 27.9 | 19.8 | 27.7 | 24.6 |
| Issue (%) | 20.7 | 28.6 | 19.6 | 27.7 | 24.6 |
| Defect (%) | 0.1 | 0.2 | 0 | 0 | 0.2 |
| Crash (%) | 26 | 14.2 | 27.9 | 9.8 | 19.1 |
| Privacy (%) | 0.3 | 1.6 | 0.2 | 0.4 | 1.9 |
| Security (%) | 2.7 | 0.2 | 4.2 | 5.7 | 0.6 |
| Spy (%) | 0 | 1.6 | 0.6 | 0 | 7.2 |
| Spam (%) | 0.6 | 1.6 | 0.2 | 1.0 | 1.4 |
| Malicious (%) | 0 | 0.2 | 0 | 0.2 | 0.6 |
| Leaks (%) | 0 | 0 | 0 | 0 | 0 |
| | | | *Most Helpful Reviews* | | |
| | Shopping | Travel | Insurance | Finance | News |
| Bug (%) | 9.5 | 8.5 | 13.5 | 10.3 | 7.6 |
| Fix (%) | 34.2 | 20.2 | 26.5 | 33.2 | 45.7 |
| Problem (%) | 23.6 | 33.2 | 20.6 | 30.6 | 21.4 |
| Issue (%) | 23.0 | 32.7 | 27.1 | 34.7 | 22.8 |
| Defect (%) | 0.1 | 0.2 | 0 | 0 | 0.3 |
| Crash (%) | 26.7 | 11.3 | 25.8 | 7.9 | 23.2 |
| Privacy (%) | 0.4 | 2.6 | 0 | 0 | 1.9 |
| Security (%) | 2.7 | 0.7 | 3.9 | 4.9 | 0.4 |
| Spy (%) | 0 | 0.7 | 0 | 0 | 0.3 |
| Spam (%) | 0.6 | 0.5 | 0 | 0.5 | 1.2 |
| Malicious (%) | 0 | 0 | 0 | 0 | 0.3 |
| Leaks (%) | 0 | 0 | 0 | 0 | 0 |

### *5.3. Comparison of Different Ranking Schemes*

Five different kinds of ranking schemes are devised using the outcome of our experiments. These are as follows: (i) ranking based on internal view, (ii) ranking based on external view, (iii) E-SERS ranking by combing the internal and external views, (iv) ranking based on average star ratings, and (v) Google Play Store Rank, (https://www.appbrain.com/stats/google-play-rankings/top/free/application/us#types, accessed on 19 March 2021).

We illustrate different scenarios for comparing these ranking schemes like our approach described in [19]. The rank-orders differ from one scheme to another; therefore, we conducted an empirical analysis to identify the reasons behind this behavior. Table 9 shows Kendall Tau distances for four such comparisons from five distinct categories.

**Table 9.** Comparing different ranking schemes.

| App Category | Average Ratings and Indirect Trust | Average Ratings and Direct Trust | Average Rating and Google Play Store Rank | E-SERS Rating and Google Play Store Rank |
|---|---|---|---|---|
| Shopping | 10% | 50% | 40% | 30–50% |
| Travel | 0% | 50% | 30% | 30–50% |
| Insurance | 40% | 60% | 30% | 40–50% |
| Finance | 30% | 50% | 40% | 40–60% |
| News | 20% | 30% | 40% | 30–40% |

**Average Ratings and Indirect Trust:** In an ideal case, the reviews should be consistent with the average star ratings given by the users—the Kendall Tau variance, as shown in Table 9, is between 0% and 40% when we compare these two rankings. This indicates that these two rankings are reasonably similar to each other. The noticed differences could be because of the following two potential reasons: (i) for our review data set, we assigned two additional weights—review-centric reputation score and the temporal weight—whereas, in an average rating score, all reviews are treated equally; and (ii) a mismatch is frequently observed between review sentiments and associated rating scores. Hence, the star ratings are not always true representations of the corresponding review narratives.

**Average Ratings and Direct Trust:** The Kendall Tau distances between these two schemes (Table 9) is between 30% and 60%. We selected an App, *App4*, from the News category that has opposite ranks—it has a rank of 2 out of 5 based on the user ratings and a rank of 5 based on the direct trust score. For further investigation of *App4*, we collected a total of 84 reviews (3.2% reviews of the total reviews) that matched with one of the keywords mentioned in Table 8. This low number (3.2%) suggests that most of the users are not concerned about the internal features of that App. Also, among these 84 reviews, most of the reviews reported crashing of the App. During the internal evidence analysis, we found critical security vulnerabilities in this App. We noticed that the data leaks associated with App4 deal with the *Dangerous* permission access (e.g., READ PHONE STATE).

**Average Ratings and Google Play Store Rank:** The Kendall Tau distance for these two schemes (Table 9) is between 30% and 40%. Factors that influence the Google Play Store ranking are App Name, App Description, Rating and Reviews, Backlinks, In-App Purchase, Updates Downloads and Engagement, and other hidden factors [74]. However, leading App Stores do not disclose how the ranking factors are weighted. To understand the correlation between average rating scores and Google Play Store ranks, we conducted an experiment. We fetched a data set of 500 Apps from AppBrain (https://www.appbrain.com/stats/google-play-rankings/top_free/applications/us, accessed on 19 March 2021) and Google Play Store, which contained Google Play Rankings, rating scores, the number of installs, and the number of reviews. This data set is the training set for a machine learning model (XGBRegressor-https://www.datatechnotes.com/2019/06/regression-example-with xgbregressor-in.html, accessed on 19 March 2021) and is used to predict an App's rank. The outcome of this experiment indicated that the *star rating*, and *the number of reviews* have a higher importance score than *the number of installs*. Since the rating score is an influential factor for the Google Play Store ranking, the disparity between these two ranking schemes is not that high.

We selected an App, *App1*, from the Shopping category that had opposite ranking positions—it had a rank of 4 out of 5 based on the user ratings and a rank of 2 based on the Google Play Store scheme. A percentage of 45% of the total reviews for this App show a below 4 rating. On the other hand, the number of installs (more than 5 million) and the number of reviews (91,857) for this App are relatively higher than those for the others. So, these factors, when combined with other Google Play Store ranking factors, give the App a higher rank.

**E-SERS and Google Play Store Rank:** Previous sections have indicated that rankings based on partial evidence result in significantly different orderings. Thus, there is a need to combine direct and indirect trust-based evidence to provide a comprehensive ranking scheme: the E-SERS approach. As we have stated that trust is subjective and indicates a user's tolerance to risk associated with installing any App on their mobile device, we do recognize that different users may associate different levels of importance to the direct and indirect artifacts of an App. We, however, adhere to the view that the direct trust evidence provides a better reflection of the App's quality to protect private information and, thus, we have assigned 70% weight to direct trust and 30% weight to indirect trust in our experiments. However, these weights can be adjusted as users desire. As can be seen from Table 9, the distance between the E-SERS ranking and the Google Play Store ranking varies from 30% to 50% and it is lesser than other distances. A higher weight for the indirect trust score reduces the distance with the Google Play Store rank, whereas a lower weight for that trust score increases the distance.

Such a scenario is illustrated with the help of *App2* in the Shopping category. *App2* is one of the top Apps ranked by the Google Play Store. User reviews and the rating scores depict a similar scenario, where approximately 78% of reviews are rated 4 stars or above. Based on the review sentiment, 70% reviews reflect positive sentiment for *App2*. E-SERS assigns a lower rank to *App2* when it is evaluated based on direct trust-based evidence. During the internal evidence analysis, we found severe security vulnerabilities in this App. Through further investigation, we found that the data leak associated with *App2* deals with the Dangerous permission access (e.g., *ACCESS FINE LOCATION* and *ACCESS COARSE LOCATION*) and these sensitive data are written to *SMS_MMS*, thereby, again, highlighting the fact that user reviews many times fail to a grasp the real view of an App and anyone relying on only reviews or star scores may regret their selection.

## 6. Conclusions

The first threat to the validity of E-SERS is that the 25 Apps used in this experiment might not be representative of the entire App Store. To address this threat, we have made our data available at https://tinyurl.com/E-SERS, accessed on 7 March 2021. However, the E-SERS approach is independent of the number of Apps used in the study. The second threat is that static data flow detection tools require all code to be accessible for analysis. We cannot fully address the issue, as we do not have access to an App's source code. However, we used the standard tools that have been used in other research studies for the Android Apps. Third, the static code analysis tools may return false-positive warnings. To overcome this limitation, we have considered the reputation score of the tools. Finally, E-SERS considered, based on a small informal survey, the top two influencing factors (Average Rating and the User Reviews) and ignored the other factors (such as Number of installs, App size, and Developer info) in the trust computations. A larger survey sample may result in different top factors and, thus, may change the trust computations. Finally, our goal is to educate and empower users in their App selection ("caveat emptor") and not "censor their freedom of choice" by filtering or de-ranking Apps. It will be a user's responsibility to select an App, hopefully based on the trust score that we are providing about that App. We do recognize that some users may either choose to ignore the trust score or may not be able to comprehend its importance. As can be seen from the Figure 5, the users are presented with all the trust scores (i.e., direct, indirect, and E-SERS-based) and it is up to them to use these details in their App selection.

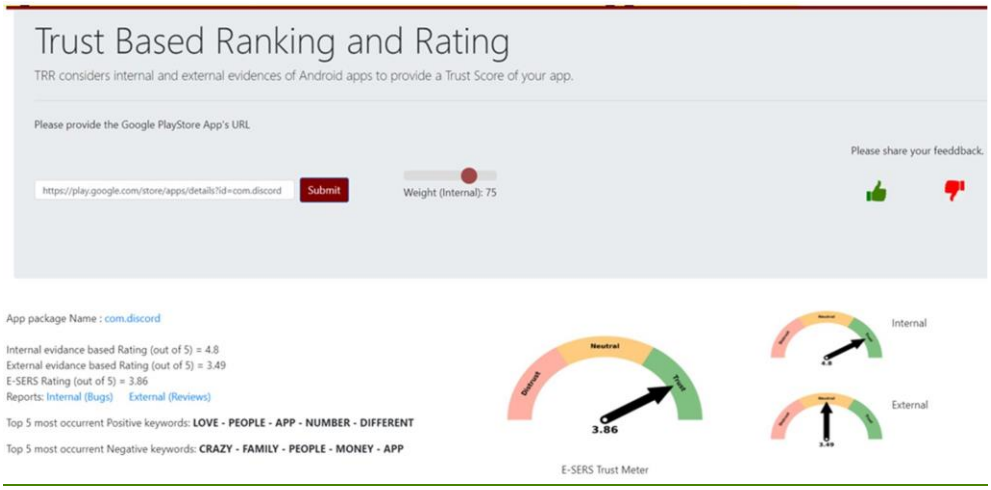



**Figure 5.** E-SERS web prototype.

We recognize that Apps-related cybersecurity is a vast topic with many facets. Our focus here is rather narrow, related to providing a holistic view of an App that considers internal and external factors of an App and its role in ranking similar Apps. We are of the opinion that such a view, as it considers multiple evidence, will empower the users to make a proper selection out of the available choices for their specific needs.

This paper has proposed a ranking scheme, called E-SERS, which is an enhanced version of our past work. These enhancements are based on formalism, a quantified risk assessment matrix, temporal weights, and reputation of reviews, as well as the incorporation of the outcome of practitioner surveys. E-SERS computes direct trust and indirect trust scores for an App using internal and external evidence and aggregates the results using subjective logic operations. The rank-ordering of similar Apps from the Google Play Store, generated by E-SERS, is based on more comprehensive analysis than prevalent alternatives. E-SERS, using the direct trust artifacts, mitigates the limitation of the presence of few reviews associated with newly published Apps and, hence, is useful to the developers, users, and society.

**Author Contributions:** Resources, A.M.; Writing—original draft, N.C.; Writing—review & editing, R.R.R. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data are contained within the article.

**Acknowledgments:** N. Chowdhury was supported by the Department of Computer and Information Science and A. Maharjan was supported by a University Fellowship at Indiana University—Purdue University Indianapolis during this research.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. How Rating Affects Ranking in Search Results and Top Charts across Platforms. 2012. Available online: https://www.adweek.com/digital/how-rating-affects-ranking-in-search-results-and-top-charts-across-platforms/ (accessed on 4 March 2021).
2. Harman, M.; Jia, Y.; Zhang, Y. App store mining and analysis: MSR for app stores. In Proceedings of the 9th IEEE Working Conference on Mining Software Repositories, Zurich, Switzerland, 2–3 June 2012.
3. Pagano, D.; Maalej, W. User Feedback in the AppStore: An Empirical Study. In Proceedings of the 21st IEEE International Requirements Engineering Conference, Rio de Janeiro, Brazil, 15–19 July 2013.
4. Svedic, Z. The Effect of Informational Signals on Mobile Apps Sales Ranks across the Globe. Ph.D. Thesis, Simon Fraser University, Burnaby, BC, Canada, 2015.
5. Martin, W.; Sarro, F.; Jia, Y.; Zhang, Y.; Harman, M. A Survey of App Store Analysis for Software Engineering. *IEEE Trans. Softw. Eng.* **2016**, *43*, 817–847. [CrossRef]

6. Finkelstein, A.; Harman, M.; Jia, Y.; Martin, W.; Sarro, F.; Zhang, Y. Investigating the relationship between price, rating, and popularity in the Blackberry World App Store. *Inf. Softw. Technol.* **2017**, *87*, 119–139. [CrossRef]
7. Lim, S.L.; Bentley, P.J.; Kanakam, N.; Ishikawa, F.; Honiden, S. Investigating Country Differences in Mobile App User Behavior and Challenges for Software Engineering. *IEEE Trans. Softw. Eng.* **2014**, *41*, 40–64. [CrossRef]
8. Martens, D.; Maalej, W. Towards understanding and detecting fake reviews in app stores. *Empir. Softw. Eng.* **2019**, *24*, 3316–3355. [CrossRef]
9. Siegler, M. YouTube Comes To A 5-Star Realization: Its Ratings Are Useless. *Techcrunch* **2009**. Available online: https://techcrunch.com/2009/09/22/youtube-comes-to-a-5-star-realization-its-ratings-are-useless/ (accessed on 4 March 2021).
10. Chowdhury, N.S.; Raje, R.R. Disparity between the Programmatic Views and the User Perceptions of Mobile Apps. In Proceedings of the 20th International Conference of Computer and Information Technology, Dhaka, Bangladesh, 22–24 December 2017.
11. Dellinger, A. Many Popular Android Apps Leak Sensitive Data, Leaving Millions of Consumers at Risk, Technical Report Forbes. 2019. Available online: https://www.forbes.com/sites/ajdellinger/2019/06/07/many-popular-android-apps-leak-sensitive-data-leaving-millions-of-consumers-at-risk/#7bc629d0521e (accessed on 4 March 2021).
12. Doevan, Android Virus, The List of Infected Apps for 2019, 2-Spyware. Available online: https://www.2-spyware.com/remove-android-virus.html (accessed on 4 March 2021).
13. Venkat, A. Kaspersky: Malware Found Hiding in Popular Android App, Bankinfosecurity. Available online: https://www.bankinfosecurity.com/kaspersky-malware-found-hiding-in-popular-android-app-a-13008 (accessed on 4 March 2021).
14. Kan, M. Malware Discovered in Popular Android App Cam-Scanner, PCMag. Available online: https://www.pcmag.com/news/malware-discovered-in-popular-android-app-camscanner (accessed on 4 March 2021).
15. Liam, T. Android Google Play App with 100 Million Downloads Starts to Deliver Malware, ZDNet. Available online: https://www.zdnet.com/article/android-google-play-app-with-100-million-downloads-starts-to-deliver-malware/ (accessed on 4 March 2021).
16. Doffman, Z. Android Warning: Devious Malware Found Inside 34 Apps Already Installed by 100M+ Users, Forbes. Available online: https://www.forbes.com/sites/zakdoffman/2019/08/13/android-users-have-installed-dangerous-new-malware-from-google-play/5c7ed4cd22a9 (accessed on 4 March 2021).
17. Chowdhury, N.; Raje, R. A Holistic Ranking Scheme for Apps. In Proceedings of the 21st International Conference of Computer and Information Technology, Dhaka, Bangladesh, 21–23 December 2018.
18. Chowdhury, N.; Raje, R. SERS: A Security-related and Evidence-based Ranking Scheme for Mobile Apps. In Proceedings of the the First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications, Los Angeles, CA, USA, 12–14 December 2019.
19. Shafer, G. *A Mathematical Theory of Evidence, White-Paper*; Princeton University Press: Princeton, NJ, USA, 1976.
20. Jøsang, A.; Hayward, R.; Pope, S. Trust network analysis with subjective logic. In Proceedings of the 29th Australasian Computer Science Conference, Hobart, TAS, Australia, 16–19 January 2006.
21. Jøsang, A.; Hayward, R.; Pope, S. A Logic for Uncertain Probabilities. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* **2001**, *9*, 279–311. [CrossRef]
22. Zhuang, L.; Jing, F.; Zhu, X. Movie review mining and summarization. In Proceedings of the 15th ACM International Conference on Information and Knowledge Management, Arlington, VA, USA, 6–11 November 2006.
23. Tang, H.; Tan, S.; Cheng, X. A survey on sentiment detection of reviews. *Expert Syst. Appl.* **2009**, *36*, 10760–10773. [CrossRef]
24. Pang, B.; Lee, L. Opinion Mining and Sentiment Analysis. *Found. Trends Inf. Retr.* **2008**, *2*, 1–135. [CrossRef]
25. Panichella, S.; Sorboy, A.; Guzmanz, E.; Visaggioy, C.; Canforay, G.; Gall, H. How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution. In Proceedings of the International Conference on Software Maintenance and Evolution, Bremen, Germany, 29 September–1 October 2015.
26. Pang, B.; Lee, L.; Vaithyanathan, S. ThumbsUp? Sentiment Classification using Machine Learning Techniques. In Proceedings of the Empirical Methods in Natural Language Processing, 6 July 2002.
27. Sangani, C.; Ananthanarayanan, S. Sentiment Analysis of App Store Reviews. *Methodology* **2013**, *4*, 153–162.
28. Palomba, F.; Linares-Vásquez, M.; Bavota, G.; Oliveto, R.; Di Penta, M.; Poshyvanyk, D.; De Lucia, A. User Reviews Matter! Tracking Crowdscourced Reviews to Support Evolution of Successful Apps. In Proceedings of the International Conference on Software Maintenance and Evolution, Bremen, Germany, 29 September–1 October 2015.
29. Gallege, L. Trust-Based Service Selection and Recommendation for Online Software Marketplaces (TruSStReMark). Ph.D. Thesis, Purdue University, West Lafayette, IN, USA, 2016.
30. Gallege, L.; Raje, R. Parallel Methods for Evidence and Trust-Based Selection and Recommendation of Software Apps from Online Marketplaces. In Proceedings of the 12th Annual Cyber and Information Security Research Conference, Oak Ridge, TN, USA, 4–6 April 2017.
31. Sarma, B.P.; Li, N.; Gates, C.; Potharaju, R.; Nita-Rotaru, C.; Molloy, I. Android permissions: A perspective combining risks and benefits. In Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, Newark, NJ, USA, 20–22 June 2012.
32. Zhou, Y.; Wang, Z.; Zhou, W.; Jiang, X. Hey, you, get off of my market: Detecting malicious Apps in official and alternative android markets. *NDSS* **2012**, *25*, 50–52.

33. Wang, Y.; Zheng, J.; Sun, C.; Mukkamala, S. Quantitative Security Risk Assessment of Android Permissions and Applications. In Proceedings of the 27th Data and Applications Security and Privacy, Newark, NJ, USA, 15–17 July 2013.
34. Gates, C.; Li, N.; Peng, H.; Sarma, B.; Qi, Y.; Potharaju, R.; Rotaru, C.N.; Molloy, I. Generating summary risk scores for mobile applications. *IEEE Trans. Dependable Secur. Comput.* **2014**, *11*, 238–251. [CrossRef]
35. Acar, Y.; Backes, M.; Bugiel, S.; Fahl, S.; McDaniel, P.; Smith, M. SoK: Lessons Learned from Android Security Research for Applied Software Platforms. In Proceedings of the 2016 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 22–26 May 2016.
36. Mirzaei, O.; Suarez-Tangil, G.; Fuentes, J. TriFlow: Triaging Android Applications Using Speculative Information Flows. In Proceedings of the ACM Asia Conference on Computer and Communications Security, Abu Dhabi, United Arab Emirates, 2–6 April 2017.
37. Arzt, S.; Rasthofer, S.; Bodden, C.F.E.; Bartel, A.; Klein, J.; Traon, Y.; Octeau, D.; McDaniel, P. FlowDroid: Precise Context, Flow, Field, Object-Sensitive and Lifecycle-Aware Taint Analysis for Android Apps. In Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation, Edinburgh, UK, 9–11 June 2014.
38. Enck, W.; Gilbert, P.; Han, S.; Tendulkar, V.; Chun, B.; Cox, L.; Jung, J.; McDaniel, P.; Sheth, A. Taintdroid: An information-flow tracking system for real time privacy monitoring on smart phones. *ACM Trans. Comput. Syst.* **2014**, *32*, 1–29. [CrossRef]
39. Gibler, C.; Crussell, J.; Erickson, J.; Chen, H. AndroidLeaks: Automatically detecting potential privacy leaks in android applications on a large scale. In Proceedings of the International Conference on Trust and Trustworthy Computing, Vienna, Austria, 13–15 June 2012.
40. Gordon, M.; Kim, D.; Perkins, J.; Gilham, L.; Nguyen, N.; Rinard, M. Information flow analysis of android applications in DroidSafe. *NDSS* **2015**, *15*, 110.
41. Cheng, X.; Luo, Y.; Gui, Q. Research on Trust Management Model of Wireless Sensor Networks. In Proceedings of the IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference, Chongqing, China, 12–14 October 2018.
42. Awan, K.; Din, I.U.; Almogren, A.; Guizani, M.; Altameem, A.; Jadoon, S. Robust Trust—APro-Privacy Robust Distributed Trust Management Mechanism for Internet of Things. *IEEE Access* **2019**, *7*, 62095–62106. [CrossRef]
43. Ruan, Y.; Zhang, P.; Alfantoukh, L.; Durresi, A. Measurement Theory-Based Trust Management Framework for Online Social Communities. *ACM Trans. Internet Technol.* **2017**, *17*, 1–24. [CrossRef]
44. Tang, T.; Winoto, P.; Niu, X. I-TRUST: Investigating trust between users and agents in a multi-agent portfolio management system, Electronic Commerce Research and Applications. *Electron. Commer. Res. Appl.* **2003**, *2*, 302–314.
45. Tang, J.; Hu, X.; Chang, Y.; Liu, H. Predictability of Distrust with Interaction Data. In Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, Shanghai, China, 3–7 November 2014.
46. Gallege, L.S.; Gamage, D.U.; Hill, J.H.; Raje, R.R. Understanding the trust of software-intensive distributed systems. *Concurr. Comput. Pr. Exp.* **2015**, *28*, 114–143. [CrossRef]
47. Jøsang, A. Artificial Reasoning with Subjective Logic. In Proceedings of the Second Australian Workshop on Commonsense Reasoning, Perth, Australia, December 1997; Australian Computer Society: Sydney, NSW, Australia. Available online: https://folk.universitetetioslo.no/josang/papers/Jos1997-AWCR.pdf (accessed on 1 July 2024).
48. Hernandez, N.; Recabarren, R.; Carbunar, B.; Ahmed, S.I. RacketStore: Measurements of ASO deception in Google play via mobile and app usage. In Proceedings of the 21st ACM Internet Measurement Conference, Virtual Event, 2–4 November 2021.
49. Farooqi, S.; Feal, Á.; Lauinger, T.; McCoy, D.; Shafiq, Z.; Vallina-Rodriguez, N. Understanding Incentivized Mobile App Installs on Google Play Store. In Proceedings of the ACM Internet Measurement Conference, Virtual Event, 27–29 October 2020.
50. Zhu, H.; Xiong, H.; Ge, Y.; Chen, E. Mobile App Recommendations with Security and Privacy Awareness. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014.
51. Cen, L.; Kong, D.; Jin, H.; Si, L. Mobile App Security Risk Assessment: A Crowdsourcing Ranking Approach. In Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, BC, Canada, 30 April–2 May 2015.
52. Jøsang, A.; McAnally, D. Multiplication and co-multiplication of beliefs. *Int. J. Approx. Reason.* **2005**, *38*, 19–51. [CrossRef]
53. Zhou, H.; Shi, W.; Liang, Z.; Liang, B. Using new fusion operations to improve trust expressiveness of subjective logic. *Wuhan Univ. J. Nat. Sci.* **2011**, *16*, 376–382. [CrossRef]
54. Skoric, B.; Zannone, N. Flow-based reputation with uncertainty: Evidence-Based Subjective Logic. *Int. J. Inf. Secur.* **2016**, *15*, 381–402. [CrossRef]
55. Avdiienko, V.; Kuznetsov, K.; Gorla, A.; Zeller, A.; Arzt, S.; Rasthofer, S.; Bodden, E. Mining Apps for Abnormal Usage of Sensitive Data. In Proceedings of the IEEE/ACM 37th IEEE International Conference on Software Engineering, Florence, Italy, 16–24 May 2015.
56. Arzt, S.; Rasthofer, S.; Bodden, E. *SuSi: A Tool for the Fully Automated Classification and Categorization of Android Sources and Sinks*; Technical Report; Technische Universität Darmstadt & Fraunhofer SIT: Darmstadt, Germany, 2013.
57. Stoneburner, G.; Goguen, A.; Feringa, A. Risk management guide for information technology systems. *Natl. Inst. Stand. Technol. (NIST)* **2002**, *800*, 800–830.
58. Joh, H.; Malaiya, Y. Defining and assessing quantitative security risk measures using vulnerability lifecycle and cvss metrics. In Proceedings of the 2011 International Conference on Security and Management, Las Vegas, NV, USA, 18–21 July 2011.

59. Android, Android Permissions Overview. Available online: https://developer.android.com/guide/topics/permissions/overview (accessed on 6 March 2021).
60. Au, K.; Zhou, Y.; Huang, Z.; Lie, D. PScout: Analyzing the Android Permission Specification. In Proceedings of the ACM Conference on Computer and Communications Security, Raleigh, NC, USA, 16–18 October 2012.
61. DroidBench—Benchmarks. Available online: https://blogs.unipaderborn.de/sse/tools/droidbench/ (accessed on 6 March 2021).
62. Shung, K. Accuracy, Precision, Recall or F1? Towards Data Science. Available online: https://towardsdatascience.com/accuracy-precision-recall-or-f1331fb37c5cb9 (accessed on 6 March 2021).
63. Brian, R. Test of 250 Popular Android Mobile Apps Reveals That 70% Leak Sensitive Personal Data; Technical Report; NowSecure: 2019. Available online: https://www.nowsecure.com/blog/2019/06/06/test-of-250-popular-android-mobile-apps-reveal-that-70-leak-sensitive-personal-data/ (accessed on 7 March 2021).
64. Unicodedata. Available online: https://docs.python.org/2/library/unicodedata.html (accessed on 6 March 2021).
65. The IBM Watson Natural Language Understanding. Available online: https://cloud.ibm.com/docs/services/natural-language understanding?topic=natural-language-understanding-getting-started (accessed on 6 March 2021).
66. Krishni, A Beginners Guide to Random Forest Regression, Medium. Available online: https://medium.com/datadriveninvestor/random-forest-regression-871bc9a25eb (accessed on 6 March 2021).
67. Jindal, N.; Liu, B. Analyzing and Detecting Review Spam. In Proceedings of the Seventh IEEE International Conference on Data Mining, Omaha, NE, USA, 28–31 October 2007.
68. Song, Y.; Wu, C.; Zhu, S.; Wang, H. A Machine Learning Based Approach for Mobile App Rating Manipulation Detection. *ICST Trans. Secur. Saf.* **2019**, *5*, e3. [CrossRef]
69. MacKay, D. *Information Theory, Inference and Learning Algorithms*; Cambridge University Press: Cambridge, UK, 2003.
70. Hawkes, A. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* **1971**, *58*, 83–90. [CrossRef]
71. Johnson, D. *Using Weights in the Analysis of Survey Data*; Population Research Institute, The Pennsylvania State University: University Park, PA, USA, 2008.
72. Kendall, M. A New Measure of Rank Correlation. *Biometrika* **1938**, *30*, 81–93. [CrossRef]
73. Maalej, W.; Nabil, H. Bug report, feature request, or simply praise? On automatically classifying app reviews. In Proceedings of the 2015 IEEE 23rd International Requirements Engineering Conference (RE), Ottawa, ON, Canada, 24–28 August 2015.
74. App Radar, App Ranking Factors How to Improve App Store Search Rankings. Available online: https://appradar.com/academy/bonus-chapters/appstore-ranking-factors/ (accessed on 7 March 2021).


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.