# A Metric for the Evaluation and Comparison of Keylogger Performance

Tobias Fiebig[λ], Janis Danisevskis[¶], Marta Piekarska[§]
*Technische Universität Berlin*
*FG Security in Telecommunications*
*Berlin, Germany*
{*tfiebig*[λ],*janis*[¶],*marta*[§]}*@sec.t-labs.tu-berlin.de*

## Abstract

In the field of IT security the development of Proof of Concept (PoC) implementations is a commonly accepted method to determine the exploitability of an identified weakness. Most security issues provide a rather straightforwad method of asserting the PoCs efficiency. That is, it either works or it does not. Hence, data gathering and exfiltration techniques usually remain in a position where the viability has to be empirically verified. One of these cases are mobile device keyloggers, which only recently have been starting to exploit side-channels to infer heuristic information on a user's input. With this introduction of side channels exploiting heuristic information the performance of a keylogger may no longer be described with "it works and gathered what was typed". Instead, the viability of the keylogger has to be assessed based on various typing speeds, user input styles and many metrics more as documented in this paper. The authors of this document provide a survey of the required metrics and features. Furthermore, they have developed a framework to assess the performance of a keylogger. This paper provides the documentation on how such a study can be conducted, while the required source code is shared online.

## 1 Introduction

Mobile devices are one of the technological developments that have profoundly changed our society. Nowadays mobile devices are used for nearly all aspects of life, including purchases, banking and communication with friends and family. Although these features are exceedingly useful, one may not wish to see this private communication exposed to somebody else or the credentials used for banking obtained by an attacker.

Programs for stealing such sensitive data are known to exist in the PC world for nearly as long as such data existed itself. Smartphones, however, usually lack the prerequisites of the simplest way for accessing key-press data, which is accessing the input stream of an attached keyboard, simply because they lack a physical keyboard. Instead these devices use a software keyboard, which is presented on the integrated touchscreen. In the past, various methods have been created that still enabled an attacker to obtain this data. These methods range from the first touchscreen based keyloggers, which read out the touch information in a manner similar to a PC's keyboard buffer [1, 5, 15, 16], over advanced systems utilizing sensory data on the mobile device to heuristically determining the user's input [2, 3, 17, 19].

If a research group discovers a new angle to obtain such information, it has to be accurately shown that the identified method is applicable and provides sufficient information on the user's keystrokes. Hence, a metric for assessing keyloggers has to include the character semantics of text. It has to go beyond simple throughput measurements of a discovered side-channel. Although many keyloggers have been implemented in the past, in each and every one of these works, distinct metrics were created and different features measured. The authors of this document recently discovered a minor flaw in a graphics library that allows for the creation of a keylogger. In order to evaluate it, they designed appropriate experiments and metrics applicable not only to the case at hand.

**Contribution to the field:** For the implemented keylogger, the authors unified and developed a set of general metrics that allow researchers to verify and prove the applicability and performance of a newly created keylogger. The presented approach does not only consider issues on a system engineering level, but also takes the subjects of such a study into account. The document at hand provides a clear documentation on how to verify and compare keyloggers. Along with the document, the developed sourcecode for such a study and subsequent data analysis is provided online. It is the authors' intent to promote further discussion and research on the empirical verification of keyloggers.

## 2 Background

As mentioned in the introduction, keyloggers are a well-known phenomenon. While it has been considerably easy to verify a keylogger in the past, as input was directly captured and could hence be directly matched to the actual input [1, 5, 15, 16], heuristic keyloggers have surfaced in the last years, which cannot be evaluated so easily. Such keyloggers exploit sensory information providing a side-channel to the actual input. As the direct access to system input is usually better protected on smartphones than on the PC platform, these keyloggers are much more common on mobile platforms than on PCs. Examples from recent years include the accelerometer-based, hence heuristic, keyloggers proposed by various authors [1, 5, 15, 16].

Another approach described in 2013 by Simon and Anderson utilizes the camera of a device to infer its relative movement, and hence the pressed keys [17]. Recent work by Lin et al. reads a smartphones screen to infer the typed keys based on the visual feedback created by the input [14]. Despite this direct method of obtaining input information, this approach still depends on the sampling rate of the keylogger. With the work of Xu et al. [18], a further method of keylogging was proposed that does not even take place on the victim's device, but instead leverages opportunities for the attacker to position optical gear in a way that allows the capture of reflections of the user's screen.

All of the latest publications have in common that they do not share a common metric, which would allow the reader to numerically compare the performance of the implemented keyloggers. While Xu et al. constructed a metric for the evaluation of the captured text, yet they refrain from using that for the evaluation of the logged passwords and instead evaluate how many guesses for a password would be needed with the gathered additional information [18]. Lin et al. follow a similar approach, focusing on how often the password has to be observed by their keylogger until a full capture has been achieved [14]. Conversly Anderson and Simon decided to evaluate the certainty with which an input could be identified [17].

Another critical aspect of these works is the methodology used for the experimental verification of the keyloggers. While in all three works experiments with subjects have been performed, none of them has actually documented how the input provided by the subjects has been verified. However, such a documentary step is necessary to allow other researchers the verification of the published results.

## 3 Method

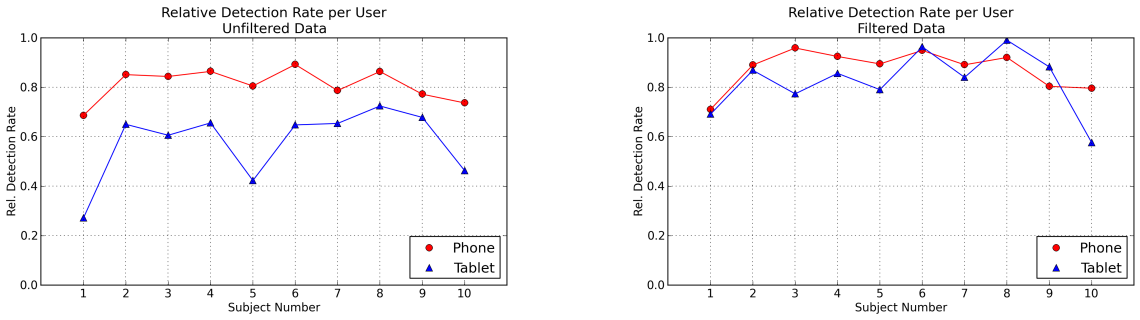
As discussed in the previous section, various features of a keylogger have to be evaluated. As the authors are focusing on keyloggers for mobile devices, the assesment of power consumption and the CPU fingerprint is important. However, the CPU fingerprint is most reliably recovered with a dedicated application running in the background of the system, to ensure that the performance of the keylogger is not impaired by this measurement. The same holds for the power consumption measurement. This measurements should ideally take place in hardware, fully transparent and out-of-band to the software on the device.

The experiment has to evaluate the performance of the keylogger based on the targeted keystroke information, i.e. it has to be evaluated on a full text containing letters, numbers and special characters. Furthermore, the accuracy of a keylogger is most critical if passwords are concerned. Hence, such should be evaluated as well, i.e. contained in the test set. The subjects have to be taken into account as well as the authors cannot expect the subjects to not mistype parts of the text they are intended to enter, leading to corresponding correction actions of the subject. The typing speed of these subjects may also be relevant for a keylogger if it relies on a side-channel that has to be measured with a specific frequency.

If all of these constraints have been taken into account, a metric has to be chosen to make the actually typed strings comparable to those recorded by the keylogger. One might argue that pre-existing work already used in the process of evaluating keyloggers might be admissable, for example the METEOR metric Lavie and Denkowski described in 2009 [12]. This metric, however, is designed for a semantic comparison between texts in different languages and not for the comparison of a text with a version of itself that possibly suffered from something most similar to transmission errors.

For fulfilling the requirements on the input mentioned in the introduction of this section, the authors created a demo text, which can be found in Appendix A. This text was specifically tailored to fulfill the previously postulated requirements. Two strings have been included that can be considered "a good password", i.e. contain special characters, numbers and characters in lower and upper case. Regular text is included as well to test the plain text recognition capabilities of the keylogger. Furthermore, providing a preset text to the subjects increases comparability as the subjects are not inclined to randomly type on the keyboard, which may lead to unreasonably high typing speeds.

To gather a complete dataset, an Android application was designed that presents the user with a text input field.




(a) (b)

Figure 1: Overall results for each of the ten subjects for both devices. (a) Without any filters applied to the dataset. One can see the similarity in the two curves. (b) With the appropriate filters applied.

An in-application logging mechanism was implemented to exfiltrate the printable character key presses to a data collection system. An instance of a demo keylogger similar to the one developed by Lin et al. [14] was running in the background of the application, extracting its findings of printable characters to another data collection system. The subjects were then instructed to enter the text, which can be found in Appendix A, on both devices subsequently while their actions were recorded in the previously presented manner. For this demo implementation, data exfiltration was done via a local wireless network connection. This was deemed acceptable for the method at hand, and no packet loss was experienced during the conducted tests. If necessary, the data collection could also be implemented on the smartphone itself.

To numerically compare the text recorded by the keylogger with the actual input text, a metric was developed based on the Damerau-Levenshtein distance as defined by Damerau and Levenshtein in 1964 and 1966 respectively [4, 13]. The Damerau-Levenshtein distance asserts a numerical value $d$ to two strings $str_1$ and $str_2$, representing their similarity by counting the number of single character operations ($o_x$) necessary to transform one string into the other. The applicable operations in the pure Levenshtein distance are deletion ($o_d$), insertion ($o_i$) and substitution ($o_s$), while Damerau also included transposition ($o_t$) as a fourth operation to be performed on characters of a string. The final formula for the Damerau-Levenshtein distance is therefore[1]: $dl(str_1, str_2) = \sum o_d + \sum o_i + \sum o_s + \sum o_t$

The lower bound of the Damerau-Levenshtein distance is 0 for two identical strings and the upper bound the number of characters in the larger string ($|characters(string)|$) for a comparison between two strings where an operation has to be performed for each character of the larger string. As this value is obviously highly dependent on the length of the two strings supplied to the function, the authors decided to modify this metric by dividing the computed Damerau-Levenshtein distance for a correctly recorded string and keylogger produced string by the number of characters in the latter. This means that the resulting metric represents the number of edits per character in the keylogger produced string. To make this value more tractable, it is represented as $(1 - value)$. Hence, larger values for the relative detection rate presented in the next section are better as they indicate a lower amount of necessary edits per character. This means that the final definition of the relative detection rate is as follows:

**Definition 1:** The relative detection rate $dt$ of a keylogging process $p$ with a typed input string *real* and a captured string *captured* is:

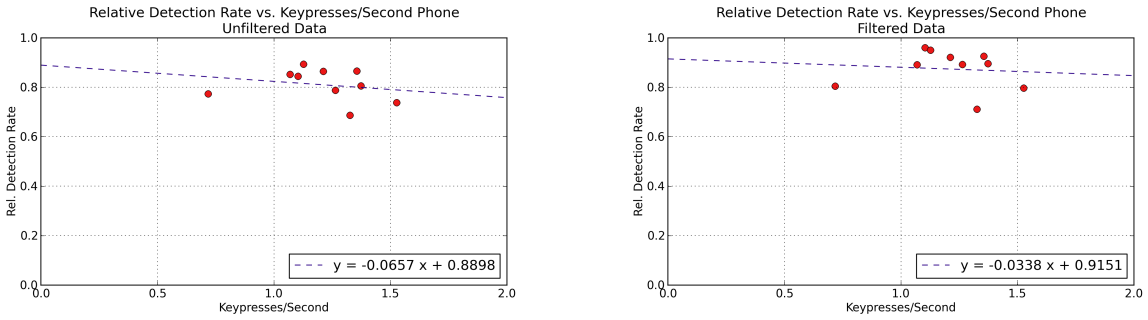
$$dt(p) = 1 - \frac{dl(real, captured)}{|characters(captured)|}$$

For example, if one were to compare a hypothetical $str_1$ = **Test String.** with a corresponding keylogger produced string $str_2$ = **Tesd tSrng.**, the Damerau-Levenshtein distance would be 3 as the **d** in *Tesd* would have to be substituted by **t**, **t** and **S** in *tSrng* would have to be swapped and an **i** added to obtain *String*. The relative detection rate would then be $1 - \frac{3}{11} = 0.73$ for the count of characters in $|str_2| = 11$ and the Damerau-Levenshtein distance of 3 between $str_1$ and $str_2$.

| Device | Android Version | Screen Size | PPI |
|---|---|---|---|
| Phone | 4.0.4 | 4.8” | 306ppi |
| Tablet | 4.1.2 | 10.1” | 149ppi |

Table 1: The software versions, screen size and PPI (Pixels per Inch) of the devices used in the experiments.

[1]Please note that this is a simplified form aiming at comprehensibility. Please consult [4] for a formally accurate version.




(a) (b)

Figure 2: Plots of the relative detection rate versus the typing speed of each subject in the phone experiments. (a) For the unfiltered dataset. Please note the low correlation between typing speed and detection rate. (b) For the filtered dataset. Note the lower correlation between typing speed and detection rate compared to Figure 2(a).

## 4 Case Study

To demonstrate the admissability of the presented method, the authors evaluated the actions of ten different subjects in a small case study on a demo keylogger. The demo keylogger exploited an issue in a memory management library widely used for GPU (Graphics Processing Unit) related memory allocation tasks. The discovered issue[2] allows unprivileged users to read the contents of memory areas holding bitmaps used by the GPU, leading to an output based keylogger similar to the keylogger of Lin et al. [14]. This is done by requesting those memory areas via a 24bit ID, with the first being one and subsequent IDs being the next free one, incrementing by one. Among those bitmaps, the on-screen keyboard of an affected smartphone can be found. Due to the visual feedback, the actual input characters can be extracted from those bitmaps. The specific implementation details are out of scope for this work and will be released in a dedicated white paper.

However, this approach is not precise as the only way to find the correct memory areas is iterating over all IDs and analyzing the corresponding memory contents. Thereby a race condition is created. The information on a key-press action can be stored in an area that is not reached by the iterative process before the memory area is de-allocated, which leads to undersampling. Hence, the iterative process may be to quick, depending on the performance of the used hardware, and read the same memory area for the same keypress twice, which means oversampling. This means that different configurations of the same keylogger have higher or lower sampling rates, depending on the hardware they run on and the softkeyboard that is used by the end user.

The authors used this feature of the demo keylogger to evaluate the comparison capabilities of the presented metric. They conducted a direct comparison between the same keylogger running on two different devices with different hardware performance. Due to the presented characteristics of the keylogger, this is similar to evaluate two different keyloggers on the same hardware. Furthermore, it was investigated how the proposed metric can be used to detect subject specific variables that influence a keylogger's performance. During this study, the typing speed of the subjects was evaluated. For the case at hand, these different test runs consisted of comparing the performance on a tablet device and a mobile phone. The devices utilized for the experiments, their operating system software version and screen size can be found in Table 1. The presented test case also demonstrated how and why data may have to be filtered during a study.

The obtained data can be found in Table 3. As can be seen the results are considerably distinct between the two keyloggers. The relative detection rate on the tablet seems to be generally lower than on the phone. This is also illustrated in Figure 1(a), which provides a direct comparison of the obtained results for each subject. The dataset for subject 1 can be found in Appendix B.

This allowed a brief investigation of the discrepancy between the experiments performed on these two devices. Two main causes were found. First, the utilized

| | |
|---|---|
| Unfiltered: | `Thhisiisademotexxt.` |
| Actual: | `This is a demo text.` |
| Filtered: | `Thisisademotext.` |
| Actual Filtered: | `Thisisademotext.` |

Table 2: A comparison between un-filtered, filtered and actual input data for subject 7 on the tablet. Please note the regular repetition of characters.

[2]The issue has been communicated to the vendor and was subsequently resolved.

| Filtered | Device | Metric | Subject | | | | | | | | | | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| No | Phone | Keys/s | 1.33 | 1.07 | 1.10 | 1.36 | 1.37 | 1.13 | 1.26 | 1.21 | 0.72 | 1.53 | 1.17 |
| | | Rel. det. rate | 0.69 | 0.85 | 0.84 | 0.86 | 0.80 | 0.89 | 0.79 | 0.86 | 0.77 | 0.74 | 0.81 |
| | Tablet | Keys/s | 1.24 | 1.19 | 1.16 | 1.12 | 1.05 | 0.95 | 1.55 | 1.19 | 1.00 | 1.27 | 1.21 |
| | | Rel. det. rate | 0.27 | 0.65 | 0.61 | 0.66 | 0.42 | 0.65 | 0.65 | 0.72 | 0.68 | 0.46 | 0.58 |
| Yes | Phone | Keys/s | 1.33 | 1.07 | 1.10 | 1.36 | 1.37 | 1.13 | 1.26 | 1.21 | 0.72 | 1.53 | 1.17 |
| | | Rel. det. rate | 0.71 | 0.89 | 0.96 | 0.92 | 0.89 | 0.95 | 0.89 | 0.92 | 0.80 | 0.80 | 0.87 |
| | Tablet | Keys/s | 1.24 | 1.19 | 1.16 | 1.12 | 1.05 | 0.95 | 1.55 | 1.19 | 1.00 | 1.27 | 1.21 |
| | | Rel. det. rate | 0.69 | 0.87 | 0.77 | 0.86 | 0.79 | 0.96 | 0.84 | 0.99 | 0.88 | 0.58 | 0.82 |

Table 3: The collected results of all experiments that have been performed. Rounded to two decimal places, averages calculated prior to rounding.

demo keylogger was unable to obtain information on the use of the space bar on the tablet. Thus, for each space key recorded by the text verification method, an edit had to be performed, heavily impairing the average detection rate. Moreover, it was found that the timing of the keylogger seems to be off on the tablet. In this case the result is an over-capturing of data. As noted in line one of Table 2, the captures of the tablet exhibit regular repetitions of characters. In contrast, the keylogger on the phone shows under-capturing, as documented in Appendix B.2.1 in comparison to B.2.2.

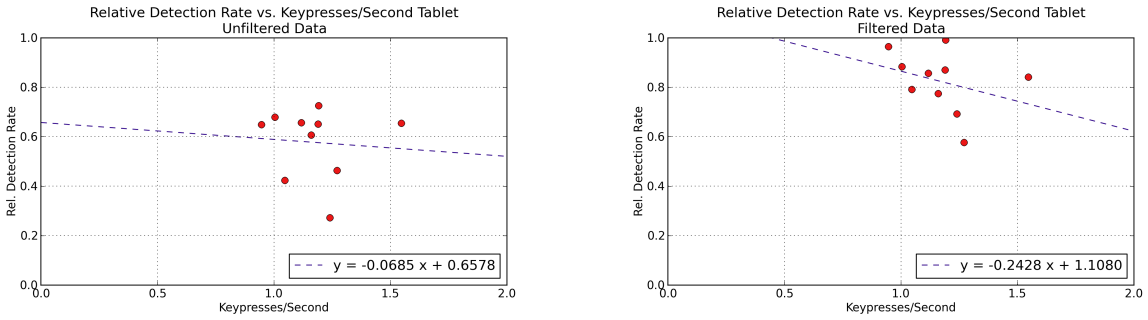
Although this demonstrates that the proposed metric can detect differences in keyloggers effectively, it is also important to verify if the obtained data can be transformed into something "more useful". Hence the data was analyzed twice. In the first step, the dataset was analyzed as-is and a plain comparison was performed. In the second step, the data was filtered to allow an evaluation of the results without being biased by the indicated timing issues. This means that immediate repetitions of the same letter were removed from the validation string as well as the keylogger produced string. For the tablet data, spaces were removed as well. Table 2 depicts the effect of this filtering on a small part of the data gathered for subject 7 on the tablet device. The exploration of the gathered data underlines that the keylogger performs reasonably well. The average detection rate for the phone is around 0.81 whereas the filtered data even yields an average detection rate of 0.87.

Although the average detection rate for the tablet is around 0.58 for the unfiltered dataset, which means that the keylogger recorded incorrect data for nearly every second character being typed, the application of the previously presented filters indicates that the data is mostly biased due to the previously mentioned effects on the tablet. On the filtered dataset, the average detection rate rises to 0.82, which surpasses the average detection rate on the phone for the unfiltered dataset, and nearly reaches the performance of the phone based keylogger on the filtered data set.

Based on the data presented in Figure 1(a), the authors assumed a correlation between a subject's typing speed, i.e. the characters per second a subject would produce, and the detection rate. If the keylogger does perform well, no such correlation should exist. For the unfiltered data of the phone experiments, a correlation factor of $r = -0.0657$ was found, as documented in Figure 2(a). This already indicates no significant correlation. The correlation coefficient of the filtered data, as documented in Figure 2(b), with $r = -0.0338$ is even smaller. Although this means that the typing speed does not have a significant influence on the detection rate, the small decrease in $r$ due to filtering yields a window of improvement in the sampling rate of the keylogger.

For the experiments performed on the tablet, a low correlation coefficient of $r = -0.0685$ can be observed for the unfiltered data set as recorded in Figure 3(a). However, as previously discussed, the filtered data for the tablet represents the actual keylogger performance more accurately than the plain data set. For the filtered dataset, a correlation coefficient of $r = -0.2428$ was found, as displayed in Figure 3(b). This lines up with the previously mentioned timing issues. However, the dataset is too sparse to allow for a reasonable statement about a causal relationship between the two variables although the authors accept this relation as an hypothesis.

The average detection rates between 0.82 and 0.87 for the filtered dataset reveal that in most cases only one edit has to be performed in approximately every 8 to 9 characters of recorded text. Considering the average length of seven to eight characters for passwords [6] and the fact that passwords are usually typed in frequently due to their purpose, this performance can be considered sufficient for gathering passwords. As far as the full text is considered, the human way of reading and comprehending text [7–10] allows an attacker to sufficiently com-




(a) (b)

Figure 3: Plots of the relative detection rate versus the typing speed of each subject in the tablet experiments. (a) For the unfiltered dataset. (b) For the filtered dataset.

prehend the captures of written full-text although characters may be switched, missing or doubled, making the method applicable for full text logging as well.

The overall result of this empirical evaluation as visualized in Figures 1(a) and 1(b) is that the keylogger performs reasonably well for extracting human readable text as well as sufficient entropy information on passwords to allow an attacker to reconstruct them.

This evaluation of the pure accuracy, however, neglects an important point for the practical applicability. If a real world attacker were to implement a keylogger, it would have to remain hidden from the legitimate user. One of the most obvious indications on something going wrong with a mobile phone, which is easily recognized by an average user, is an unusualy high CPU load, or rather the associated quick battery drainage. As Hoffmann et al. have shown in 2013 [11], the fingerprinting of power consumption is not a reliable metric to detect malware automatically. Therefore the presented accuracy metric has to be combined with a metric for the CPU consumption of a keylogger. As the use-cases of an implemented keylogger may vary, this combination of CPU usage versus the average detection rate in a specific configuration should be performed individually for each keylogger evaluation.

## 5 Discussion and Future Work

As already discussed in Section 2, no unified method of evaluation was available prior to this work. However, the presented method only constitutes the first step towards a commonly accepted standardized framework. The reader is explicitly invited to contribute to this effort by reading and improving the framework source code available online. The unification of the presented accuracy metrics with metrics for the detectability, i.e. usability fingerprint, of keyloggers is a challenge that should be approached by the community. In this context, it should be thoroughly discussed to which extent it is desirable to integrate these metrics in a holistic keylogger metric. Furthermore, the integration of semantic properties into the metric should be improved. Various psycholinguistic publications indicate that a text with a low degree of diversion from the original is still readable, this should be integrated into the metric more closely [7–10].

It would be interesting to re-evaluate previously proposed keyloggers with this unified method, providing the research community with a clear and direct comparison of what was done in the past, and how the different heuristic keylogging techniques identified on mobile devices relate to each other. This endeavour would, however, require the cooperation of the authors of the related publications as the produced source code is usually not publicly available. This raises another question, namely the habit of not publishing extensive information on the experimental setup and the code used for the implemented keylogger along with scientific work in this field. Publishing that material would not only improve the verifiability of conducted work, it would also aid aspiring researchers in understanding the subject matter at hand.

## 6 Conclusion

With the presented work, researchers now have a fully documented metric and empirical process at hand, with which the detection performance of an implemented keylogging method can be evaluated. By demonstrating the process for an example use-case, the authors have documented how this metric and surrounding experimental setup can be practically utilized. The implemented experimentation tools have been made available to the public at `https://gitlab.sec.t-labs.tu-berlin.de/mobile-keylogger/metric`. This set does not only contain the experimentation tools for the described methods, but also the necessary tools to

evaluate and graph the data gathered in an experiment. To enable researchers to quickly get a first glance at the evaluation and experimentation process, the toolkit also holds the test-data from the performed use-case study and a demo keylogger.

## Acknowledgments

This work was partially supported by the EU FP7/2007-2013 (FP7-ICT-2011.1.4 Trustworthy ICT), under grant agreement no. 317888 (project NEMESYS) and BMWF grant 01IS12032. The authors would also express their gratitude for the valuable feedback provided by the anonymous peer reviewers.

## References

[1] ARON, J. Touchscreen keylogger captures details from your taps. *New Scientist 211*, 2825 (2011), 21.

[2] AVIV, A. J., SAPP, B., BLAZE, M., AND SMITH, J. M. Practicality of accelerometer side channels on smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference* (2012), ACM, pp. 41–50.

[3] CAI, L., AND CHEN, H. On the practicality of motion based keystroke inference attack. In *Trust and Trustworthy Computing*. Springer, 2012, pp. 273–290.

[4] DAMERAU, F. J. A technique for computer detection and correction of spelling errors. *Communications of the ACM 7*, 3 (1964), 171–176.

[5] DAMOPOULOS, D., KAMBOURAKIS, G., AND GRITZALIS, S. From keyloggers to touchloggers: Take the rough with the smooth. *Computers & Security* (2012).

[6] DELL'AMICO, M., MICHIARDI, P., AND ROUDIER, Y. Password strength: an empirical analysis. In *INFOCOM, 2010 Proceedings IEEE* (2010), IEEE, pp. 1–9.

[7] DREWNOWSKI, A., AND HEALY, A. F. Detection errors on the and and: Evidence for reading units larger than the word. *Memory & Cognition 5*, 6 (1977), 636–647.

[8] GREENBERG, S. N., HEALY, A. F., KORIAT, A., AND KREINER, H. The go model: A reconsideration of the role of structural units in guiding and organizing text on line. *Psychonomic bulletin & review 11*, 3 (2004), 428–433.

[9] HEALY, A. F. Detection errors on the word the: Evidence for reading units larger than letters. *Journal of Experimental Psychology: Human Perception and Performance 2*, 2 (1976), 235.

[10] HEALY, A. F. Letter detection: A window to unitization and other cognitive processes in reading text. *Psychonomic Bulletin & Review 1*, 3 (1994), 333–344.

[11] HOFFMANN, J., NEUMANN, S., AND HOLZ, T. Mobile malware detection based on energy fingerprintsa dead end? In *Research in Attacks, Intrusions, and Defenses*. Springer, 2013, pp. 348–368.

[12] LAVIE, A., AND DENKOWSKI, M. J. The meteor metric for automatic evaluation of machine translation. *Machine translation 23*, 2-3 (2009), 105–115.

[13] LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady* (1966), vol. 10, p. 707.

[14] LIN, C.-C., LI, H., ZHOU, X., AND WANG, X. Screenmilker: How to milk your android screen for secrets. In *Proceedings of 21th USENIX Network Distributed System Security Symposium* (2014), USENIX Association.

[15] MAGGI, F., VOLPATTO, A., GASPARINI, S., BORACCHI, G., AND ZANERO, S. A fast eavesdropping attack against touchscreens. In *Information Assurance and Security (IAS), 2011 7th International Conference on* (2011), IEEE, pp. 320–325.

[16] SAGIROGLU, S., AND CANBEK, G. Keyloggers. *Technology and Society Magazine, IEEE 28*, 3 (2009), 10–17.

[17] SIMON, L., AND ANDERSON, R. Pin skimmer: Inferring pins through the camera and microphone. In *Proceedings of the 3rd ACM workshop on Security and privacy in smartphones and mobile devices* (2013), ACM.

[18] XU, Y., HEINLY, J., WHITE, A. M., MONROSE, F., AND FRAHM, J.-M. Seeing double: reconstructing obscured typed input from repeated compromising reflections. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (2013), ACM, pp. 1063–1074.

[19] XU, Z., BAI, K., AND ZHU, S. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks* (2012), ACM, pp. 113–124.

# A Demo Text Provided to Subjects

This is a demo text. The demo text includes 23 passwords. Like this one 4%23$awD or this one 98%=!awdD. Just kidding. It is only two passwords. Is it not? Please add a new line with something you would consider a secure password. No need to remember it.

# B Collected Data

Please note that due to spacial constraints this section only contains the full data for subject 1. The full dataset is available online at:
https://gitlab.sec.t-labs.tu-berlin.de/mobile-keylogger/metric

## B.1 Subject 1 - Tablet

### B.1.1 Captured

TTThhhiiiissssiiiissssaaaadedddeddeeddemootteetteexxtxxtt..%%TThheeddeeddeeddeooeeotteettxxtxxtxtiinniiccnnccllluudduuduudduudlluudduudduussdssd2233lleettettrrtrrssrrsppaasssswsswwoorroorrddss..LLLiiikkkeekkeettthhiissoonnneee444%%2233$$$$aaawwwDDDooooorrrrttttthhhhiiiissssoooonnnneee99998898998%%%====!!!aaawwwdddDDD...JJJJuuuJJsssttkkkiikkkiiiddddiiinnnggg...IIIttttiiiissssssooooonnnnllllyyytttttwwwwooooppppaaaassssswwwwooooorrrddddsss...Isssiiittttnnnooottt...PPPllleeeaaaeeeaaaeeeaeeeaaaadddaaannneeeewweewwewllliiinnneeennnwwwwiiiitthtthtttthsssosoossmmemeemmmmeettttthhhttthiiinnnggggnnnggggnnnggyyyyooouuuwwwooouuuulllldddlllddcccooonnnsssiiidddiiidddiiiidddrrraaassseeecccuuuuurrrreeeeepppppaaaaassssswwwooorrrddd...NNNoooonnnneeeddddtttooorrrreeeeemmmmeeemmmeeemnnneeeeeeddddtttooorrreeemmmeeemmmemmmbbbeeerrriiitttt....

### B.1.2 Real

This is a demo text. %The demo text inckude ludes 23 letterspasswords. Like this one 4% 23$awD or this one 98%=!awdD. Just kidding. It is noly two passwords. Is it not? Please addanew line with somethigng you would consider a secure password. Noneed toremem need to remember it.

### B.1.3 Captured Filtered

Thisisadedmotext.Thedemotextincludese23psaswordrds.Likethis4%232$awDorthisone0ne989%=!awdD.Justkikidigng.Itisonlytwopaswordrs.Isitnot.?PlPeaeaeadanewewliniewithsomethingyouwouldconsidededeasescurerpaswsword.Nonedtorerememberit.

### B.1.4 Real Filtered

Thisisademotext.Thedemotextiincludes23paswords.Likethis4%23$awDorthisoe0ne98%=!awdD.Justkidigng.Itisonlytwopaswords.Isitnot.?Pleaseadanewlinewithsomethingyouwouldconsiderasecurepasword.Nonedtorememberit.

## B.2 Subject 1 - Phone

### B.2.1 Captured

Ti  is a text.The edm tet ncdes 2 psswds.Like this onne 42awD or this one 98=!!awD..Jus  kiddininn. It s onl to pasord. Is it nt? Plaee ad new line wih sonehighng you wol considera scue aswor.No ned to remembe it..

### B.2.2 Real

This is a demo text. The edmo text includes 23 passwords.. Likrethis one 4%23$awD or this one 98%=!awdD. Just kuddingidding. It is only two passwords. Is it not? Please add a new line with sonerhingmething you would consider a secure password. No need to rememb er it.

### B.2.3 Captured Filtered

tisThis is a demo text.The demo xt inclus 23 pasword.Limw e this one 423$wD or rthis one 98%=! awdD.Just kiding. It i obnlzy two padswors. Is it no?Pleas ad a nw line wit o mthige ou would considera secure paswsword. N o ed to rememer it.

### B.2.4 Real Filtered

thisThis is a demo text. The demo text incl udes 23 paswords. Likw e this one 4%23$awD or rthis one 98%=! awdD. Just kiding. It is onlzy two padswords. Is it not? Please ad a new line with somthinge you would consider a secure paswsword. N o ned to remember it.