BrokenStrokes: On the (in)Security of Wireless Keyboards

Gabriele Olinger*, Savio Sciancalepore*, Simone Raponi*, Roberto Di Pietro*
* Division of Information and Computing Technology
College of Science and Engineering, Hamad Bin Khalifa University - Doha, Qatar
{ssciancalepore, goligeri, rdipietro}@hbku.edu.qa, {sraponi}@mail.hbku.edu.qa

Abstract—Wireless devices resorting to event-triggered communications have been proved to suffer critical privacy issues, due to the intrinsic leakage associated with radio frequency emissions.

In this paper, we move the attack frontier forward by proposing BrokenStrokes: an inexpensive, easy to implement, efficient, and effective attack able to detect the typing of a pre-defined keyword by only eavesdropping the communication channel used by the wireless keyboard. BrokenStrokes achieves its goal when the eavesdropping antenna is up to 15 meters from the target keyboard. We prove the attack succeeds regardless of the encryption scheme, the communication protocol, the presence of radio noise, and the presence of physical obstacles.

We tested BrokenStrokes in three real scenarios (close to the keyboard—e.g., the eavesdropping device is concealed under the desk—, wall separation—eavesdropping from next office—, and eavesdropping from the public street—into the house of one of the co-authors), under the following conditions: presence of radio noise, testing arbitrary long keystroke sequences, and varying several system parameters. Performance are striking: BrokenStrokes detects the presence of a keyword among the user’s keystrokes in 90% of cases when the eavesdropping antenna is placed in the proximity of the keyboard (up to 20 cm), while it guarantees at least 75% success rate when the eavesdropping antenna is up to 15 meters far away from the target. We discuss the rationale for the attack, its logical flow, and we detail the experimental setting and the algorithmic machinery adopted. Finally, we discuss potential countermeasures and sketch some future research directions.

The data utilized in this paper have been released as open-source to allow practitioners, industries, and academia to verify our claims and use them as a basis for further developments.

Index Terms—Wireless Keyboards, Wireless Security, Traffic Analysis, Event-Triggered Communications.

I. INTRODUCTION

Wireless keyboards are becoming more and more popular in homes, offices, and entertainment systems, enabling a smooth, tiny, and elegant interaction with computing devices [11]. Especially in crowded offices, wireless keyboards reduce the number of wires to be managed per working location, with evident advantages in elegance and neatness. Besides, they extend the interaction area with terminals, allowing stress-less and pain-free working experiences [2], [3].

The increasing popularity of wireless keyboards is evident when looking at the global market trends. For the time frame 2017-2025, the global market growth for wireless keyboards is estimated at a value of 3.7% per year [4]. This is due to several factors such as the integration of various features and devices, increasing traction in smart TV applications, and design innovations.

Despite their popularity, wireless keyboards suffer several confidentiality and privacy issues, mainly caused by the broadcast nature of the wireless communication link and energy constraints [2], [5]. In fact, compared to legacy wired keyboards, wireless keyboards use a wireless communication medium, where the information is inherently exposed to potential eavesdropping [6], [7]. At the same time, being powered by batteries, wireless keyboards have to implement efficient computation and communication strategies, minimizing the Radio Frequency (RF) operations to increase the lifetime of the batteries [8], [9], [10].

From the security perspective, many legacy wireless keyboards deploy very weak (or none) protection against eavesdropping attacks. In the cited context, attacks can be easily achieved by tuning a malicious receiver at the same operating frequency of the keyboard [11]. A few researchers [12], [13] also demonstrated the feasibility of active attacks, such as keystroke injection and replay, capable of poisoning the communication link and reducing the usability and security of wireless keyboards.

While manufacturers are designing, implementing and delivering more and more secure solutions for wireless keyboards, the intrinsic security of wireless keyboards has still to deal with usability and energy constraints [14]. Indeed, wireless keyboards have to trigger a new RF communication for each new keystroke, to guarantee the minimum typing delay and maximum usability. At the same time, such RF communication should last for the minimum amount of time, to minimize the battery drain and increase the lifetime of the keyboard battery itself [11], [15].

Contribution. In this paper, we present BrokenStrokes, a novel attack able to detect the presence of keywords in arbitrarily long keystroke sequences by only eavesdropping the (encrypted) keyboard-dongle communication link. The underlying strategy of BrokenStrokes is based on the identification and acquisition of Received Signal Strength (RSS) samples associated with the keystrokes of a target user. Thanks to advanced analyses carried out on the interarrival times between adjacent keystrokes, BrokenStrokes can enable a variety of attacks, including the identification of the number of keystrokes associated with a keyword, as well as the detection of a specific keyword in a stream of keystrokes. Overall, BrokenStrokes is a very inexpensive and easy-to-perform attack, requiring only
a commercial Software Defined Radio (SDR) and an antenna working on the 2.4 GHz frequency band. Moreover, BrokenStrokes is a completely agnostic attack: in fact, it requires no information about the specific physical-layer modulation scheme, MAC-layer communication protocol, packet format, and even the encryption technique used for the communication between the wireless keyboard and the receiving dongle. Further, it is resilient to radio noise.

We provide the details of BrokenStrokes, as well as the algorithms to extract the keystrokes from the noisy RSS samples and to detect the presence of a pre-defined keyword. Moreover, we tested BrokenStrokes in three reference scenarios, where the adversary is located: (i) in close proximity; (ii) behind a wall; and, (iii) at an increasing distance from the keyboard-dongle communication channel, without any further effort to reduce the noise from other devices sharing the same communication frequency. BrokenStrokes achieves outstanding detection accuracy in all the considered scenarios. For instance, in the user’s close proximity, BrokenStrokes can detect the keyword in 90% of the cases. In Line-Of-Sight (LOS) scenarios, with distances up to 15 meters, BrokenStrokes can detect the presence of a keyword in an arbitrary long sentence with a frequency of about 80%, independently of the particular distance. BrokenStrokes can also detect a keyword in 73% of the cases when the eavesdropping device is separated by the target from a wall (in an office environment). Furthermore, we provide some suggestions on countermeasures that can be implemented to contain BrokenStrokes, highlighting the trade-offs between energy consumption and security issues. Finally, the data about our attacks have been released as open-source at the link [16], to allow practitioners, industries, and academia to verify our claims and use them as a basis for further development.

Paper Organization. The remainder of this paper is organized as follows: Section II summarizes recent attacks against keyboards, while Section III illustrates the scenario we assumed for our work and initial assumptions. The intuition behind BrokenStrokes is introduced in Section IV while Section V provides the details on the methodology to detect a keyword in a sequence of keystrokes, by exploiting the RSS. The three scenarios tackled by our contribution are described in Sections VI, VII, and VIII, respectively. Section IX provides the results of our attack in all the cited scenarios, while Section X provides further details on the feasibility of BrokenStrokes, as well as some limitations. Finally, Section XI tightens conclusions and draws some future work.

II. RELATED WORK

Keylogging side channel attacks can be classified as a function of different parameters [17], including targets (user, keyboard, host or network), modality (acoustic, wired, WiFi, seismic, motion, EM radiations), and proximity (close proximity—e.g., smartwatch—, few meters, or up to 15 meters). In the following, we provide a brief overview of such attacks, collecting them according to the particular channel modality enabling the attack, i.e., motion, acoustic, communication, and seismic.

Motion. This class of attacks investigated the feasibility of keystroke eavesdropping and reconstruction via information acquired using hand-held constrained devices. In this context, the authors in [18] showed that any word typed by a user while wearing a smartwatch can be found in a shortlist of a median of 24 words. The size of this shortlist decreases as the number of characters of the target word increases, e.g., 10-words-median shortlist when the word has more than 6 characters. The proposed attack exploits the Bayesian Inference and requires training from the attacker’s end. Similarly, the authors in [19] showed that wearable devices embedded sensors can be exploited to discriminate user’s fine-grained hand movements, by leveraging hand trajectories to reconstruct the typed PINs. They implemented a training-less Backward PIN-Sequence Inference algorithm exploiting both the physical distance and the temporal interval between the keys to build a tree of candidate key entries. The technique achieved 80% accuracy with one PIN typing and more than 90% accuracy with three attempts. Other similar studies are reported by the authors in [20], where the authors reconstruct QWERTY inputs, and [21], where also PIN inputs are recovered.

Acoustic. These attacks aim at recognizing keystrokes by identifying the sound generated by pressing different keys. One of the first attacks of this kind has been proposed in [22]. The authors exploited neural network classifiers for the recognition of the keys and proposed the design of homophonic keyboards as a possible countermeasure. The authors in [23], starting from a 10-minute sound recording, built an attack able to recover up to 95% of the typed characters. The approach leverages both Machine Learning (ML) and speech recognition techniques. The authors in [24] presented a dictionary attack based on keyboard acoustic emanations. With a combination of signal processing and efficient algorithms, the approach allows reconstructing words of 7-13 characters. The attack does not require any training and allows finding the typed word in the top 50 candidates with 90% accuracy. Several years later, the authors in [25] showed the feasibility of using the microphone embedded in an off-the-shelf smartphone to discriminate millimeter-level position differences, thus enabling keystroke snooping. By relying on ML techniques, the authors achieved outstanding performance, up to the 94% of correctly identified keystrokes. A similar attack was described by the authors in [26], by exploiting the Time Difference of Arrival (TDoA) between typing activities to estimate the physical position of the keystrokes. The experiments show that more than 72.2% of keystrokes can be successfully recovered. The authors in [27] explored a new keyboard acoustic eavesdropping attack using the popular Voice-over-IP (VoIP) service Skype. Leveraging the K-nearest neighbors statistical learning algorithm, they reconstructed the keystrokes typed by a victim on a remote keyboard while involved in a Skype call, coming up with an accuracy up to 91.7%.

Communications. This class of attacks explored the possibility of reconstructing the keystrokes of target users by extracting information from the communication channel, be it wired or wireless. Focusing on the wired setting, an early analysis of keystrokes timing attacks has been provided by [28]. The authors collected inter-keystroke timings from Ethernet sessions using the Secure Shell (SSH) protocol, and they inferred on the bigrams typed by the user. The
proposed solution allows to significantly reduce the entropy of passwords transmitted as encrypted via an SSH tunnel. While being characterized by outstanding performance, this solution requires physical access to the Ethernet link. Besides, it is suitable only for reducing the complexity of single word instances, such as passwords. In the context of wireless communication attacks, the authors in [29] described the limitations of detecting compromised electromagnetic waves with a wide-band receiver tuned on a specific frequency. As a result, they proposed a new effective attack, consisting of acquiring the raw signal from the antenna and processing the entire electromagnetic spectrum. Despite being quite an expensive solution, this side-channel attack can recover 95% of keystrokes on a PS/2 keyboard, from up to 20 meters, and through walls. Similarly, the authors in [3] introduced a novel attack exploiting WiFi signals, which correlates the hand movement with text writing. When any user types a certain key, his fingers move uniquely and generate a unique pattern in the Channel State Information (CSI). The authors exploited WiFi signals to perform keystroke recognition by using two commercial devices: a sender (i.e., a router) and a receiver (i.e., a laptop). When evaluated in real-world experiments, the approach recognizes keystrokes typed to form sentences with an accuracy of 93.5%. A similar attack has been described by the authors in [31], based on the identification of the changes in the wireless channels related to a keystroke. By relying on five antennas and signal-cancellation techniques, the proposed solution reaches 91.8% accuracy with full-training and 80% accuracy with reduced training input. Eavesdropping attacks based on the channel state information extracted from wireless signals have emerged as effective strategies and can be delivered without relying on a training phase [30].

Seismic. The approaches in this class leverage one or more accelerometers to detect the vibrations generated by a nearby keyboard, and use these pieces of information to infer the specific keystroke activity. In this context, the authors in [31] showed that mobile phones equipped with an accelerometer can reconstruct text typed on a nearby keyboard by only considering the vibrations perceived by the device. The application proposed by the authors allows to detect and decode the keystroke by measuring both the distance between each vibration and the physical position of the devices involved. The experiments showed that the proposed approach could recover approximately 80% of the typed content. The vibrations have been exploited also by [32], where the proposed attack can detect and recognize words based on the mechanical vibration produced by keystrokes.

Others. Besides the categories listed above, the literature includes other attacks aiming at undermining either the security or the privacy of the information typed on a keyboard by unaware users. Among the interesting approaches, we can find those based on videos [33], [34], [35], [36], [37], [38], [39], CPU [40], [41], [42], [43], and Memory [44], [45], [46], [47].

Compared to the above valuable approaches, BrokenStrokes emerges as a very flexible attack, agnostic of any communication protocol implemented for the keyboard-dongle communication, being effective from 20cm up to 15m, and requiring minimal and cost-effective equipment.

III. SCENARIO AND ASSUMPTIONS

We consider a general scenario constituted by a wireless keyboard system, i.e., a keyboard transmitting wirelessly the user’s keystrokes to a USB dongle connected to a computer. In this scenario, our attack affects all the wireless communication protocols that could be employed to sustain the communication between the keyboard and the dongle, such as Bluetooth, WiFi, and proprietary ones. Without loss of generality, we consider three widely adopted wireless keyboards, as depicted in Table I. All of them feature proprietary communication protocols exploiting the ISM bandwidth $[2.4 \text{– } 2.5]$ GHz for the communication. We stress that our solution involves neither the hacking nor the reverse engineering of the protocols adopted by the considered wireless keyboards.

TABLE I: Keyboards Brands and Models considered in our analysis.

| Brand       | Model       | Frequency range [GHz] | Protocol   |
|-------------|-------------|-----------------------|------------|
| HP          | SK-2064     | $[2.4 \text{– } 2.5]$ | Proprietary|
| Microsoft   | 850-1455    | $[2.4 \text{– } 2.6]$ | Proprietary|
| V-MAX       | K-201       | $[2.4 \text{– } 2.6]$ | Proprietary|

Equipment. Table II summarizes the equipment adopted throughout this paper. We adopted a commercial laptop (Dell XPS 15 9560), featuring a Linux distribution and GNU Radio (a free and open-source software development toolkit), a commercial Software Defined Radio (SDR) [48], and either an omnidirectional (VERT2450) or a directional antenna (Aaronia HyperLOG 60350), depending on the considered attack scenarios. Finally, all the proposed algorithms, techniques, and procedures adopted throughout this paper have been implemented in Matlab R2019a.

TABLE II: Equipment.

| Type          | Description                      |
|---------------|----------------------------------|
| Laptop        | Dell XPS 15 9560                 |
| Antenna       | VERT2450 Omni-directional antenna|
| SDR           | MiriadRF LimeSDR                 |

Scenario. We performed extensive measurements of the BrokenStrokes attack in the following reference scenarios:

1) Scenario 1: Proximity attack. The SDR features a standard omnidirectional antenna (VERT2450). We placed the SDR in the close proximity of the keyboard-dongle communication link—we concealed it under the desk. This attack involves the adversary having access to the location of the target user (e.g., office, home), and being able to place the SDR very close to the wireless keyboard, e.g., under the user’s desk or in its close proximity.

2) Scenario 2: Behind-the-wall attack. The SDR is connected to a directional antenna (Aaronia HyperLOG 60350) and there is no LOS between the antenna and the keyboard-dongle communication link. This attack takes into account an adversary willing to collect the inter-keystrokes timings of a target user while being behind obstructing objects, such as walls [49], [50], thus possibly remaining undetected.
3) **Scenario 3: Remote attack.** In our setting the SDR is connected to a long-range directional antenna (Aaronia HyperLOG 60350), the adversary is located far away from the target user, but he has a clear LOS to the target and he can collect the inter-keystroke timings from a remote location (up to 15m).

**Multiple Users.** We considered three different users, namely \{U_1, U_2, U_3\}, and we evaluated how the user’s typing pace affects the BrokenStrokes attack. The number of users adopted in this paper is consistent with related works on keystrokes analysis [23, 26, 31].

**Noise.** Finally, we remark that our measurement campaign has been executed in regular office conditions, without any effort to reduce the noise generated by other devices sharing the same communication frequency of the target keyboards.

### IV. BrokenStrokes IN A NUTSHELL

Figure 1 illustrates the block diagram and the core components of the **BrokenStrokes** attack.

![BrokenStrokes: System architecture and components.](image)

The computing flow of **BrokenStrokes** is composed of: (i) measuring the Received Signal Strength (RSS) of the messages transmitted between the keyboard and the dongle; (ii) exploiting such measurements to extract inter-keystroke timings; and, finally, (iii) resorting to a Machine Learning (ML) technique to generate a likelihood score, indicating the presence of a pre-defined keyword in the keystroke sequence of the target user.

We adopted the MiriadRF LimeSDR to measure the RSS of the packets generated by each keystroke event [51], and we resort to GNU Radio to tune the parameters of the SDR [52]. Specifically, we observed that wireless keyboards are idle when no keystrokes are typed. As soon as the user presses any button, a new transmission from the keyboard to the dongle is triggered, generating a peak at a specific operating frequency. Figure 2 shows the GNU Radio block diagram to fulfill the measurement of the RSS associated with the keyboard-dongle communication link. As an example, we observe that the RF Frequency in the LimeSuite Source (RX) block is the one adopted by the HP SK-2064 wireless keyboard.

We connected the LimeSDR Source (RX) standard module, configured with a proper operating frequency, a bandwidth of 10 MHz, and a sample rate of 30 MHz, to a QT GUI Frequency Plot module, where we enabled the log of the RSS (in dBm) and a timestamp (in nanoseconds), when the value of the RSS on the particular operating frequency exceeds a predefined threshold value. Figure 3 shows a screen-shot of the acquisition of one sample from the GNURadio software. The blue line shows the RSS associated with each frequency in the pre-defined range, while the yellow line shows the output of the maximum hold function. The peak at -16 dBm corresponds to a transmission between the keyboard and the dongle.

![GNU Radio frequency view: the blue line shows the RSS associated with each frequency in the pre-defined range, while the yellow line shows the output of the maximum hold function. The peak at -16 dBm corresponds to a transmission between the keyboard and the dongle.](image)

In this section, we show the details of the **BrokenStrokes** attack, providing the mechanisms that can be used by an adversary to detect the presence of a keyword in an arbitrarily...
long sentence typed by a user through a wireless keyboard. Without loss of generality, we consider the Scenario 1, i.e., the Proximity attack, while in the later sections, we will extend our methodology to the other scenarios.

Figure 4 shows the RSS samples collected from the SDR with a sampling rate of $30^6$ samples per second. We asked $U1$ to type 50 times the keyword “password”, and we collected the RSS estimations associated with the messages exchanged between the keyboard and the dongle. The RSS samples show a clear pattern, consisting of vertical bands: one band per word, since $U1$ was typing a keyword, hitting return, and then starting again—for a total of 50 repetitions of the word “password”. The solid red line in Fig. 4 shows the threshold we used for filtering RSS samples, i.e., only the samples above the threshold are considered for the subsequent processing. The importance of the threshold will be clear later on, when filtering out the samples associated with interference while retaining only the samples coming from the keyboard-dongle communication. Indeed, we observe that, in this specific scenario (Proximity attack), the vast majority of the samples are mainly concentrated in the range of $[-20, -35]$ dBm and, therefore, any threshold less than -35dBm can be adopted for this purpose.

In the following, we extract the inter-keystroke timings via a dual-stage process: (i) Words Identification; and, (ii) Keystroke Timings Extraction. The first phase exploits RSS samples to identify the words typed by the target user, while the second phase focuses on extracting the inter-keystroke timings associated with the word previously identified.

A. Words identification

Let us reconsider the samples collected for the experiment of Fig. 4, and the RSS threshold set at 50 dBm. As previously stated, we filter out all the RSS samples with intensity less than -50dBm, and we only consider the ones above the RSS threshold. Moreover, to the aim of the inter-keystroke timings extraction process, the absolute intensity values of the RSS are not relevant. Therefore, we normalize all of them to the same value, still preserving the time, as depicted by the circles (one per RSS sample) in the top part of Fig. 5.

To extract the timings associated with the beginning of each word, we considered a sliding window of a pre-determined duration, and we count for the number of samples belonging to it (when sliding from the beginning to the end of the trace). The sliding-window size is important and its configuration depends on both the user and the word to be detected. As an example, for the word “password”, we considered sliding windows of size 2.4, 1.7, and 2 seconds, for the user $U1$, $U2$, and $U3$, respectively. Moreover, we empirically assumed a sliding step of 1/50 of the window size. Finally, in this work, we assume that the sliding-window duration can be pre-set by the adversary. Indeed, it can properly calibrate the sliding window by looking at the collected samples, and set it up accordingly. The bottom part of Fig. 5 shows the number of samples belonging to the sliding window, given a certain delay (in milliseconds) from the beginning of the trace. At the same time, the peaks in the bottom part of Fig. 4 represent the beginning of a new word. Indeed, if the sliding window duration is properly calibrated, the number of samples is the highest possible when the window is at the beginning of the word, while it is less than the maximum when the window is randomly placed elsewhere in the trace. Vertical red lines in the top part of Fig. 5 show the identified peaks in relation to the RSS sample positions (black circles).

B. Keystroke timings extraction

Recalling Fig. 5 we observe that each word is mainly constituted by several samples in the range between 250 and 300. In the following, we provide the details of the methodology adopted to extract the keystroke timings from the RSS samples, as well as how much these timings can be considered as a good approximation of the real keystroke timings. For each of the identified words (vertical red lines in Fig. 5), we performed the following analysis. Firstly, we focused on the samples collected from a single word, as depicted in the top part of Fig. 5. We observe that the word “password” is constituted by 9 groups of samples (8 letters and the carriage
return). Each group of samples, in turn, can be divided into two sub-groups: the first set of about 20 samples and the second set of about 5 samples, as depicted in the bottom part of Fig. 6.

To correctly identify the keystroke timings, we adopted a sliding window duration of 0.024 seconds and a sliding step of 1/50 of the window size. The sliding window duration takes into account the communication round-trip-delay between the keyboard and the dongle and, being dependent on the keyboard brand/model, it requires a pre-processing of the collected samples. The above parameters have been optimized for the HP SK-2064, while we will discuss the impact of the keyboard hardware on the performance of the BrokenStrokes attack in a later section of this paper (Sec. X).

Finally, by considering the peaks from the previous analysis, we identified the keystroke timings as depicted by the vertical red lines in the top part of Fig. 6.

![Keystroke samples (word “password”)](image)

Fig. 6: Keystroke timings extraction: we count for the number of keystroke timings collected during 50 repetitions of the word “password” using the keylogger. In the previous analysis, we did not take into account the carriage-return keystroke, but only the timings between two subsequent keystrokes within the word “password”. We highlight that we considered only the quantile 0.05 of the keystroke interarrival times, since it represents the worst case, i.e., the keystroke pairs with the 5% smallest time difference. The top part of Fig. 7 shows the absolute value of the difference (error) between the inter-keystroke timings collected by the BrokenStrokes attack and the ones collected by adopting the keylogger (bottom part of Fig. 7). For each box, the central mark represents the median, while the bottom and top edges of the box indicate the 25-th and 75-th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the red ‘+’ symbol.

We observe that, even in the worst-case scenario, the error is always less than 20ms, compared to an average inter-keystroke timing of 200 ms, computed over the data collected by the key logger. To sum up, we highlight that the median value of the error is about 5ms (for all the users), being the 2% of the quantile 0.05 of the inter-keystroke timings collected by the key logger.

**User independence.** We stress that Words Identification and Keystroke Timings Extraction are independent of the user, i.e., the processing performed by the SDR introduces only a minor delay that does not affect the pattern of the inter-keystroke timings. Therefore, already proposed techniques that affect user’s privacy by exploiting inter-keystroke timings, such as the one in [17] and [28], can be significantly enhanced by moving the adversary far away from the target user. In the following, we propose a ML-based technique to estimate the likelihood associated with the position of a pre-defined keyword inside a sentence.

### C. Keyword detection

Inter-keystroke timings have already been adopted in the literature to infer on patterns and words typed by a target user [17]. Nevertheless, to the best of our knowledge, no one has proposed so far to extract inter-keystroke timings from RSS samples. Moreover, our attack significantly improves the chances of the adversary to remain undetected during the guessing procedure. Nevertheless, the combination of the attack peculiarities and the adopted scenarios require a different methodology compared to the ones already proposed in the literature; in particular, we propose a ML-based solution that is resilient to both small inter-keystroke timings errors and interference experienced during the eavesdropping phase.

We considered a Support Vector Machine (SVM) classifier trained with only one class (one-class SVM classifier), i.e., 50 instances of the word “password”. Our intuition is to discriminate the keyword “password” from outliers (other words) by resorting to a likelihood score computed by the SVM classifier. The keyword detection phase is performed by the ML module of the BrokenStrokes attack (recall Fig. 1) and consists of the following steps:

1) **Training.** We trained a one-class SVM model with 50 replicas of the keyword “password”. We adopted
We define a Decision Threshold, and the related statistical metrics, i.e., True Positives, and False Positives.

**Definition.** Let \( \{s_0, \ldots, s_N\} \) be a set of similarity scores. We define Decision Threshold (\( \Delta \)) as the similarity score value such that \( \min_i(s_i) + \Delta \) represents the minimum value to assume the keyword as included in the sentence.

**Definition.** We define True Positives the similarity scores that exceed \( \Delta \) and, at the same time, feature a position (offset) consistent with the actual position of the keyword in the current sentence.

**Definition.** We define False Positives the similarity scores that exceed \( \Delta \) and that, at the same time, feature a position (offset) not consistent with the actual position of the keyword. We assume a position as not consistent when its distance from the actual beginning of the keyword is larger than two keystrokes.

In the next sections we consider \( \Delta = 0 \) (i.e., we do not consider the effect of \( \Delta \)), while in Section \( \text{VX} \) we will study the performance of BrokenStrokes for different values of \( \Delta \).

It is worth noting that the above-described procedure does not require the attacker to know the time when the specific keyword is typed. Indeed, the attacker can first acquire all the keystrokes, and then perform the attack.

**VI. Scenario 1: Proximity Attack**

We estimate the performance of BrokenStrokes in a real-world scenario. We ask user U1 to repeat 30 times three different sentences: (i) your password is secret; (ii) the secret of your password; and, (iii) your secret password is mine, being these sentences characterized by the presence of the keyword at different offsets from the beginning of the sentence. We considered Scenario 1 (Proximity attack), and therefore we placed the eavesdropping equipment very close to the keyboard-dongle communication link, in a regular office scenario, with people moving around and several sources of interference, i.e., many WiFi networks and Bluetooth devices. The scope of the attack is two-fold: (i) detecting the presence of the word “password” in the sentences; and, (ii) providing the likelihood of its whiting the keystroke stream. Given the proximity between the Software Defined Radio and the keyboard-dongle communication link, we adopted the standard VERT2450 omnidirectional antenna (recall Table I), directly connected to the SDR.

The BrokenStrokes attack is very easy to deploy in this scenario, given the portability of the MiriadRF LimeSDR. The SDR can be installed under the office desk and powered up by a standard USB power bank. This attack assumes a very close distance between the eavesdropping antenna and the keyboard-dongle communication link; during our measurements, we have considered a distance between 10 and 20 cm.

Figure 8 shows the similarity scores provided by the SVM classifier as a function of the sliding window offset. The sliding window duration has been calibrated on the number of inter-keystroke timings, i.e., 7, constituting the keyword “password” while the sliding step is equal to one keystroke. A peak in the similarity score at a certain offset means that the subsequent samples are likely to match with the samples in the training set, and therefore, the current offset is likely to be the beginning of the keyword.
We observe that, for all the three sentences, the SVM classifier returns higher similarly scores at the offset where the keyword “password” begins. Moreover, we observe that the BrokenStrokes attack can locate the position of the password, while also experiencing a certain level of uncertainty, i.e., not all the major peaks are located exactly at the position where the keyword begins. Indeed, recalling Section V interference can either add fake keystrokes or make the existing ones not retrievable. Therefore, there are cases where the keyword detection occurs earlier, i.e., some of the keystrokes have not been retrieved, or later, i.e., fake keystrokes have been generated by environmental interference. Overall, this phenomenon just slightly affects the performance of our attack, and the uncertainty of the key position is usually in the range of ±1 keystroke from the actual position. By reconsidering the results from Fig. 8, we extracted the maximum score for each sentence and we compared its position with the one corresponding to the actual position associated with the beginning of the keyword “password”. Figure 9 shows the number of occurrences as a function of the error in computing the expected position of the keyword. We observe that about 31% of the detection events do not suffer from any error (27 out of 90). Moreover, we observe that 45% of the detection events are affected by an error of just one keystroke, while a mere 14% of the detection events occur 2 keystrokes earlier than the real one. Therefore, BrokenStrokes can locate the keyword “password” in 90% of the cases with an error of less than 2 keystrokes. A solid red line in Fig. 9 shows the best fit distribution being a normal distribution with mean -1.06 and standard deviation 2.47.

VII. SCENARIO 2: KEYWORD DETECTION FROM BEHIND A WALL

In Scenario 2, we perform the BrokenStrokes attack in an environment characterized by crowded neighboring offices, setting up the eavesdropping equipment in one office and launching the attack from the neighboring office. The target user was aware of our attack and collaborated with us when asked to repeat 30 times the same sentence, i.e., you can choose a random password. The antenna has been placed 4.5 meters away from the target user, while a concrete wall of about 20 cm was obstructing the Line-Of-Sight. This attack is particularly powerful since it does not require the adversary to gain access to the premises of the target user; therefore, it is particularly suitable to compromising air gap systems, where the security of one or more computers is guaranteed by physically isolating them from insecure environments.

The attack has been performed using the Aaronia HyperLOG 60350 directional antenna connected to a LimeSDR and, in turn, to a laptop, as previously described in Section III. Moreover, as for the previous cases, we converted the RSS samples to inter-keystroke timings, we trained an SVM model with 10 repetitions of the keyword “password”, and we tested the model on subsequent inter-keystroke timings extracted from the collected logs. As for the previous cases, we used 7 subsequent samples (sliding window), i.e., the number of inter-keystroke timings of the keyword “password”. The test procedure returns a similarity score for each sliding window, and we report such scores in Fig. 10 as a function of the sliding window offset. We repeated the previous procedure for a sequence of 30 sentences containing the keyword “password” at the 25th keystroke. We observe that the vast majority of the similarity score peaks are concentrated at offsets 24 and 25, i.e., the lag of one keystroke is mainly due to lost samples during the eavesdropping phase. Moreover, we highlight the presence of peaks far away from the expected offset, i.e., one at 19 and a few more in the range from 7 to 13. We consider these peaks as false positives, i.e., the keyword is not present, but our algorithm still estimated its presence as likely. Nevertheless, in 19 cases out of 30, the algorithm correctly identifies the position of the keyword, while in 10 cases BrokenStrokes provides a (slightly) wrong position for the keyword.

The number of false positives is mainly due to two factors: (i) the wall obstructing the Line-Of-Sight affects the RSS of the samples transmitted by the keyboard; and, (ii) the office environment is particularly prone to interference. BrokenStrokes is particularly sensitive to interference since it exploits RSS estimations to generate the inter-keystroke timings. Therefore, WiFi access points, Bluetooth devices, or any other radio device transmitting at the same frequency of the keyboard might affect the performance of our attack. In such cases, additional filtering techniques might be adopted but, to ease the discussion, in this work we do not consider them. Nevertheless, we will discuss in detail the strategies to mitigate the number of false positives in the next sections.

We notice that, despite the tested sentences all contain the keyword under test password, the very low similarity score levels in the positions where the keyword is not present provide an indication of the very high robustness of the BrokenStrokes attack to false positives. In fact, there is no word achieving the same levels of similarity scores as the ones obtained when the keyword is present.

VIII. SCENARIO 3: REMOTE ATTACK

In this section, we consider Scenario 3 (Remote Attack), where the adversary leverages a directional antenna (Aaronia HyperLOG 60350) to perform the BrokenStrokes attack. In this scenario, the target user sits at the ground floor of a two floors villa in [anonymized for peer-reviewing], in the proximity of a window. We placed the eavesdropping antenna at 1, 5, 10, and 15, and 20 meters from the keyboard-dongle communication link. We stress that the link between the directional antenna and the keyboard-dongle is obstructed by only a window, and therefore we consider it as a LOS attack.

Figure 11 shows the RSS samples associated with the distances previously considered. Firstly, we observe that the interference significantly increases when the eavesdropping antenna moves away from the target user (black area at the bottom of the figures). This effect can be explained by observing that the main lobe of the directional antenna becomes more and more exposed to transmitting entities that might be located in the neighborhood villas, e.g., WiFi, Bluetooth, and other interfering sources. Moreover, we observe that the peaks associated with the actual RSS samples belonging
Fig. 8: Keyword detection inside a sentence: we tested the BrokenStrokes attack against three different sentences (repeated 30 times each) by searching for the keyword “password”. Similarity scores are generated by the SVM classifier and the peaks represent the likelihood for the beginning of the keyword “password”.

Fig. 9: Frequency of the errors associated with the prediction of the keyword position.

Fig. 10: Behind-the-wall attack scenario: BrokenStrokes attack is performed against the target keyboard being separated from the eavesdropping antenna by an office wall to the keyboard-dongle communication channel are varying between -15dBm and -20dBm at 1m and 20m, respectively. Finally, we calibrated the thresholds (horizontal red lines), by empirically considering the lowest possible values with minimum interference.

Interfering samples affect the performance of BrokenStrokes: in the following, we measure the performance of the two core components of BrokenStrokes, i.e., Word Identification and Keystroke Timings Extraction, previously introduced in Section V-A and Section V-B respectively. We run the previously discussed techniques for all the collected measures at a distance of 1, 5, 10, 15, and 20 meters. Figure 12 (top) shows the results of our analysis: the BrokenStrokes attack can identify about 100% of the words up to a distance of 10 meters, while its performance decreases to about 54% and 24% at 15 and 20 meters, respectively. Moreover, Figure 12 (bottom) shows that BrokenStrokes can successfully retrieve 9 out of 9 keystrokes (“password” + carriage return) up to 10 meters, while interference significantly affects performance starting at a distance of 15 meters. Nevertheless, we observe that the number of extracted keystrokes is still high, even at a distance of 20 meters, with a median value of 8 keystrokes identified out of 9.

Word Identification and Keystroke Timing Extraction are not enough to detect the presence of the keyword in the keystrokes of the target user. Therefore, in the following, we apply a ML technique (SVM) to compute the likelihood (similarity score) of the presence (and position) of the keyword “password” in a sentence typed by a remote target user, as previously described in Section IV. Figure 13 shows the similarity scores generated by the SVM algorithm trained with 10 repetitions of the keyword “password”. Each similarity score is computed by testing a sliding window of 7 inter-keystroke timings with a moving step of 1 keystroke.

Table III shows the number of True Positives and False Positives (out of 30 sentences) as a function of the eavesdropping distance. BrokenStrokes can detect the presence and the position of the keyword in the vast majority of the cases (≥73%), i.e., the peaks of the similarity scores are concentrated at about the same offset (±1) of the actual position of the keyword.
(a) 1 meter  (b) 5 meters  (c) 10 meters  (d) 15 meters  (e) 20 meters

Fig. 11: Received Signal Strength (RSS) at 1m, 5m, 10m, 15m, and 20m (from left to right) and related thresholds (red lines) to filter out the noise.

Fig. 12: Word extraction ratio (out of 50 repetitions of “password”) and number of extracted keystrokes (out of 9), with increasing distance. Error bars show quantiles 0.05, 0.5, and 0.95 associated with the number of keystrokes per word extracted from 50 repetitions of “password”.

TABLE III: Remote attack scenario: TP Vs FP

| Distance (m) | TP    | FP    |
|-------------|-------|-------|
| 5           | 24/30 | 5/30  |
| 10          | 22/30 | 7/30  |
| 15          | 24/30 | 5/30  |

“password”. Moreover, we observe the presence of a few false positives (≤23%), i.e., there are minor peaks distributed at different offsets of the sentence. A thorough analysis of this phenomenon is provided in Section [IX].

IX. BrokenStrokes performance

In this section, we provide an estimation of the BrokenStrokes attack performance, considering all the three discussed scenarios altogether. As previously detailed, we trained a statistical learning algorithm (SVM) with a sequence of 10 repetitions of the keyword “password”, and we tested such a model on adjacent subsets (sliding windows) of several sentences. For each test, the SVM algorithm provides a similarity score (i.e., likelihood) that such a subset of characters matches the keyword we are looking for. Therefore, each sentence becomes a vector of similarity scores. The highest the score, the highest is the likelihood of that score identifying the beginning of the keyword “password”.

In previous sections, we considered a decision threshold ∆ = 0, while in the following, we study how ∆ affects the performance of BrokenStrokes—we will vary ∆ in the range [0...0.03]. Figure [14] shows the True Positives estimations as a function of ∆. We consider all the major measurements we already discussed in this paper: (i) Proximity attack (sentence “your password is secret”); (ii) Behind-The-Wall attack (sentence “you can choose a random password”); and, (iii) Remote attack at distances of 5, 10, and 15 meters (sentence “a password has many characters”), respectively. We didn’t consider 1m and 20m: the former has performance similar to Scenario 1 (Proximity Attack), while the latter one is affected by too many interferences. Firstly, we observe that the scenarios Proximity and Remote attack (all the distances) are characterized by similar trends that have been summarized by the solid green line in Equation [1]:

\[ TP = -27.522 \cdot \Delta + 0.83648, \]

where \( TP \) are the True Positives and \( \Delta \) is the Decision Threshold, respectively. Moreover, we observe that the worst performance are from the Behind-The-Wall scenario; indeed, as previously discussed such scenario is the only one deprived of Line-Of-Sight, while at the same time suffering from interference generated by neighboring devices.

Lastly, we highlight that BrokenStrokes can detect the presence of a keyword in the inter-keystrokes timings samples of a target user with a frequency of about 80%, independently of the considered scenario.

Figure [15] shows the False Positives estimations as a function of the Decision Threshold (\( \Delta \)). As for the previous case, we consider the: (i) Proximity attack (sentence: “your password is secret”); (ii) Behind-The-Wall attack (sentence: “you can choose a random password”); and, (iii) Remote attack at distances of 5, 10, and 15 meters (sentence: “a password has many characters”), respectively. Figure [15] confirms that the Behind-The-Wall scenario is the least performing: for all the thresholds, the false positives in this scenario are significantly higher than the ones in the other scenarios (although being always less than 35%). Conversely, the other scenarios show better performance, being characterized by some False positives—always less than 25%.

The Decision Threshold (\( \Delta \)) value should be chosen to maximize the number of true positives, while at the same time...
(a) 5 meters
(b) 10 meters
(c) 15 meters

Fig. 13: Detecting the keyword “password” inside a sentence for Scenario 3 (Remote attack): we tested BrokenStrokes against the same sentence (repeated 30 times) at three distances, i.e., 5 meters (a), 10 meters (b), and 15 meters (c). Similarity scores are generated by the SVM classifier and the peaks represent the likelihood for the beginning of the keyword “password”.

Fig. 14: True positives increasing the Decision Threshold ($\Delta$).

Fig. 15: False positives increasing the Decision Threshold ($\Delta$).

reducing the number of false positives. Nevertheless, given the results of Fig. [14] and Fig. [15] we observe that $\Delta$ should be chosen as small as possible ($< 5 \cdot 10^{-3}$) in order to experience high values of true positives, and therefore, low values of missed detection (false negatives). Conversely, the number of false positives can be estimated as 10% (on average) when $\Delta < 5 \cdot 10^{-3}$; we deem this values as an acceptable one, since we can assume one or more additional layers of post-processing to reduce the number of false alarms, by exploiting advanced ML techniques—though left for future work, we highlight the issue in next section. The source data adopted by this work have been released as open-source at the link [10], to allow practitioners, industries, and academia to verify our claims and use them as a basis for further development.

X. DISCUSSION, LIMITATIONS, AND COUNTERMEASURES

In the following, we discuss the importance of the training set size, some limitations of BrokenStrokes and, finally, a few potential countermeasures to mitigate its impact.

Training set size. The effectiveness of BrokenStrokes strongly relies on the training set previously collected by the adversary. On the one hand, large training sets might be difficult to collect in a reasonable amount of time, and therefore, the attack feasibility is strictly related to the number of required repetitions of the keyword to achieve good detection performance. On the other hand, small training sets can be easily collected by simple social engineering techniques, for instance triggering a response from the user (i.e., having his typing) via e-mail or social networks, to cite a few. We studied the performance of the BrokenStrokes attack with different training set sizes, from 5 to 50 repetitions of the keyword “password”. We considered the 30 repetitions of the sentence “your password is secret” from Scenario 1 (Proximity Attack) as our test set, and we run the BrokenStrokes attack as described in the previous sections. The optimal size of the training set is 10, guaranteeing the maximum number of True Positives (29) and minimizing the number of False Positives (1), as depicted in Table [IV].

| Train set size | TP  | FP |
|---------------|-----|----|
| 5             | 25  | 3  |
| 10            | 29  | 1  |
| 20            | 28  | 2  |
| 30            | 26  | 4  |
| 40            | 26  | 4  |
| 50            | 24  | 6  |

TABLE IV: True Positives (TP) and False Positives (FP) as a function of the training set size.

We highlight that the training set size leading to the best results depends on both the keyword and the typing pace of
the user. Thus, a preliminary phase is required to estimate the optimal training set size for each keyword-user combination.

**Keyboard communication protocol.** The effectiveness of *BrokenStrokes* might be affected by the communication protocol used by the wireless keyboard at the physical layer. The vast majority of keyboards adopt proprietary protocols, like the ones used throughout this paper. These protocols usually select a frequency and keep it for a long-term period (up to the switch-off or battery replacement). This behavior is particularly prone to the *BrokenStrokes* attack, since the attacker can monitor the ISM band of the wireless spectrum, in the range 2.4–2.5 GHz, identify the frequency adopted by the target user, and select that target frequency for collecting the RSS samples.

Figure 16 shows the inter-sample timings for the three different keyboards discussed in Section III. We distinguish three categories: (i) Intra-packet samples; (ii) Packet-Ack delay; and, (iii) Inter-keystroke timings. Intra-packet samples are the RSS estimations belonging to the same packet, being either the message from the keyboard to the dongle or the acknowledgment from the dongle to the keyboard. The second category (Packet-Ack delay) is the time between the packet and the ack: the keyboard Microsoft 850-1455 seems to have a very small delay compared to both HP and V-Max. We consider 16ms as the upper bound for the previous category. Finally, Inter-Keystroke timings represent the time between two consecutive keystrokes. *BrokenStrokes* is effective if and only if the user’s typing speed is lower than the Packet-Ack delay. When the user’s typing speed becomes comparable to the Packet-Ack delay, the current version of *BrokenStrokes* is not able to distinguish between the Ack of a packet and the packet associated with the subsequent keystroke. We recall that we empirically chose 23ms (Section V-B) to uniquely identify the Packet-Ack pattern for the keyboard HP SK-2064.

![Figure 16: Keyboards comparison: we considered three keyboards and compared their inter-sample timings.](image)

Some keyboard producers resort to the Bluetooth communication technology, implementing frequency hopping techniques. In these cases, a larger spectrum observation is required to capture the RSS samples of the pseudo-randomly chosen frequencies, thus increasing the cost of the equipment used to launch our attack. Moreover, some other keyboard producers (a negligible fraction of, though) adopt the Direct Sequence Spread Spectrum (DSSS) modulation, which spreads the information over a wide-band channel, significantly reducing the transmission peak power and making the communication almost indistinguishable from the noise floor.

**External interference.** The main drawback to *BrokenStrokes* is interference. As previously discussed, other devices sharing the same frequencies of the keyboard-dongle communication link might significantly affect the performance of the attack. We studied the effect of interference, by considering different parameters (i.e., RSS thresholds), equipment (i.e., directional and omnidirectional antenna), and scenarios (i.e., Proximity, Behind-The-Wall, and Outdoor). We proved that interference can be mitigated and *BrokenStrokes* can guarantee the detection of a keyword with high chances (more than 70% in the harshest conditions), independently of the configuration.

**Countermeasures.** Mitigating *BrokenStrokes* involves a few strategies: (i) increasing the number of transmissions by either beaconing or friendly jamming; (ii) randomly delaying the keyboard transmissions; or, (iii) adopting DSSS instead of fixed or pseudo-random frequency hopping techniques. The first two strategies might be impractical for wireless keyboards, since they require more energy, with the second one also possibly affecting the user experience. Wireless keyboards are mainly event-triggered devices and the trade-off between energy, usability, and privacy has already been widely investigated [15]. Finally, although DSSS might appear an effective strategy, it is more energy-consuming than frequency hopping techniques [53] and, therefore, one has still to carefully consider the trade-off between privacy and energy budget.

**XI. Conclusion and Future Work**

In this paper, we have introduced *BrokenStrokes*, a novel, inexpensive, viable, efficient, and effective attack targeting commercial wireless keyboards. *BrokenStrokes* allows to detect the presence of a pre-defined keyword in a stream of user-generated keystrokes, by just analyzing the wireless traffic generated by the keyboard.

We studied the effectiveness of *BrokenStrokes* in three different scenarios, including proximity to the target user, LOS with distances spanning between 1 and 15 meters, and non-LOS scenarios (eavesdropping from behind a wall in a crowded office environment). All the scenarios are characterized by remarkable performance even in the presence of noise (from more than 70%+ in the harshest conditions to 90%+ in normal operating conditions), confirming both the viability and effectiveness of the attack. We also highlighted some limitations of the proposal, as well as future interesting research directions.

**REFERENCES**

[1] K. Ali, A. X. Liu, W. Wang, and M. Shahzad, “Recognizing Keystrokes Using WiFi Devices,” *IEEE Journal on Selected Areas in Commun.*, vol. 35, no. 5, pp. 1175–1190, 2017.

[2] A. Boitan, R. Bărrușu, M. Popescu, B. Valerici, and O. Fratu, “Wireless Keyboards Communication Interception - The Balance Between Convenience and Security,” in *Int. Conf. on Commum. (COMM)*, 2018, pp. 539–542.
A. Compagno, M. Conti, D. Lain, and G. Tsudik, “Don’t skype & type!: Energy efficiency in wireless sensor networks: A top-down survey,” Comp. Networks, vol. 67, pp. 104–122, 2014.

G. Y. Li, Z. Xu, C. Xiong, C. Yang, S. Zhang, Y. Chen, and S. Xu, “Energy-efficient wireless Commun.: tutorial, survey, and open issues,” IEEE Wireless Commun., vol. 18, no. 6, pp. 28–35, 2011.

T. Halevi and N. Saxena, “Tracking Keystrokes Using Wireless Signals,” in Proc. of Int. Conf. of Mobile Systems, Applications, and Services, 2015, pp. 31–44.

C. Cimpanu. (2019) Fujitsu wireless keyboard model vulnerable to keystroke injection attack. [Online]. Available: https://www.zdnet.com/article/fujitsu-wireless-keyboard-model-vulnerable-to-keystroke-injection-attacks

T. Zhu, Q. Ma, S. Zhang, and Y. Liu, “Context-free attacks using keyboard acoustic emanations,” in Proc. of ACM Conf. on Comp. and Commun. Security, 2014, pp. 453–464.

A. Compagno, M. Conti, L. Dain, and G. Tsudik, “Don’t skype & type!: Acoustic eavesdropping in voice-over-ip,” in Proc. of ACM on Asia Conf. on Comp. and Commun. Security, 2017, pp. 703–715.

Y. Xu, J. Heinly, A. M. White, F. Monrose, and J. Frahm, “Seeing double: Reconstructing obscured typed input from repeated compromising reflections,” in Proc. of 13 ACM Conf. on Comp. and Commun. Security, 2013, pp. 1063–1074.

D. Zhu, R. Kumar, A. Serwadda, and V. V. Phoha, “Beware, your hands reveal your secrets!” in Proc. of ACM Conf. on Comp. and Commun. Security, 2014, pp. 904–917.

P. Marquardt, A. Verma, H. Carter, and P. Traynor, “(sp) iphone: Tracking acoustic side channel attacks: exploring realistic and security-sensitive scenarios,” Int. Journal of Information Security, vol. 14, no. 5, pp. 443–456, 2015.

P. Leu, I. Puddu, A. Ranganathan, and S. Čapkun, “I Send, Therefore I Leak: Information Leakage in Low-Power Wide Area Networks,” in Proc. of 11th ACM Conf. on Security & Privacy in Wireless and Mobile Networks, 2018, pp. 23–33.

BusinessWire. (2016) Bastille discovers “keysniffer” vulnerability in wireless keyboards which reveals private data to hackers in clear text. [Online]. Available: https://www.businesswire.com/news/home/20160726005331/en/Bastille-DisCOVERS-%E2%80%9CKeySniffer%E2%80%9D-Vulnerability-Wireless-Keyboard-REveals

A. Toutven and N. Saxena, “Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios,” Int. Journal of Information Security, vol. 14, no. 5, pp. 443–456, 2015.

F. Adib, C. Hsu, H. Mao, D. Katabi, and F. Durand, “Capturing the state-of-the-art, and challenges,” in Proc. of ACM Conf. on Comp. and Commun. Security, 2018, pp. 1063–1074.

D. X. Song, D. Wagner, and X. Tian, “Timing Analysis of Keystrokes and Timing Attacks on SSH,” in {USENIX} Security Symposium, vol. 10, 2001.

M. Vuagnoux and S. Pasini, “Compromising Electromagnetic Emanations of Wired and Wireless Keyboards,” in {USENIX} Security Symposium, vol. 13, no. 1–26, 2019, pp. 1–16.

S. Fang, I. Markwood, Y. Liu, S. Zhao, Z. Lu, and H. Zhu, “No Training Hurdles: Fast Training-Agnostic Attacks to Infer Your Typing,” in Proc. of ACM Conf. on Comp. and Commun. Security, 2018, pp. 1747–1760.

P. Marquardt, A. Verma, H. Carter, and P. Traynor, “(sp) iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers,” in Proc. of ACM Conf. on Comp. and Commun. security, 2011, pp. 551–562.

A. Barisani and D. Bianco, “Sniffing keyboards with lasers/voltmeters,” Proc. of Black Hat USA, 2009.

D. Balzarotti, M. Cova, and G. Vigna, “Clearshot: Eavesdropping on keyboard input from video,” in IEEE Symp. on Security and Privacy, 2008, pp. 170–183.

K. Ali, A. X. Liu, W. Wang, and M. Shahzad, “Keystroke recognition using wifi signals,” in Proc. of Int. Conf. on Mobile Comput. and Networking, 2015, pp. 90–102.

F. M. Insights. (2018) Global market study on wireless keyboard. [Online]. Available: https://www.persistencemarketresearch.com/report/global-market-study-wireless-keyboard-market

F. Adib, C. Hsu, H. Mao, D. Katabi, and F. Durand, “Capturing the state-of-the-art,” in Proc. of ACM Conf. on Comp. and Commun. Security, 2018, pp. 1063–1074.

J. T. Trostle, “Timing attacks against trusted path,” in Proc. of 1998 IEEE Symp. on Security and Privacy, 1998, pp. 125–134.

M. Schwarz, M. Lipp, D. Gruss, S. Weiser, C. Maurice, R. Spreitzer, and S. Mangard, “KeyDrown: Eliminating Keystroke Timing Side-Channel Attacks,” arXiv preprint arXiv:1706.06381, 2017.

J. Sun, X. Yin, Y. Chen, J. Zhang, Y. Zhang, and R. Zhang, “VISIBILITY: Video-Assisted Keystroke Inference from Tablet Backside Motion,” in Network and Distributed System Security Symp., 2016,

D. Balzarotti, M. Cova, and G. Vigna, “Clearshot: Eavesdropping on wifi signals,” in Proc. of Int. Conf. on Mobile Comput. and Commun. Security, 2015, pp. 90–102.

R. Akeela and B. Dezfouli, “Software-defined Radios: Architecture, Evolution, and Challenges,” in Proc. of 20th USENIX Security Symp., 2011, pp. 1063–1074.

S. Fang, I. Markwood, Y. Liu, S. Zhao, Z. Lu, and H. Zhu, “No Training Hurdles: Fast Training-Agnostic Attacks to Infer Your Typing,” in Proc. of ACM Conf. on Comp. and Commun. Security, 2018, pp. 1747–1760.

P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, “[DRAMA]: Exploiting [DRAM] Addressing for Cross-CPU Attacks,” in USENIX Security Symp., 2015, pp. 897–912.

G. Baldini, T. Sturman, A. R. Biswas, R. Leschhorn, G. Godor, and M. Street, “Security aspects in software defined radio and cognitive radio networks: A survey and a way ahead,” IEEE Commun. Surveys & Tuts., vol. 14, no. 2, pp. 355–379, 2012.

F. Adib, C. Hsu, H. Hao, D. Katabi, and F. Durand, “Capturing the Human Figure Through a Wall,” ACM Trans. Graph., vol. 34, no. 6, pp. 1–13, 2015.

F. Adib and D. Katabi, “See Through Walls with WiFi!” SIGCOMM Comput. Commun. Rev., vol. 43, no. 4, pp. 75–86, 2013.

R. Akeela and B. Dezfouli, “Software-defined Radios: Architecture, state-of-the-art, and challenges,” Comp. Commun., vol. 128, pp. 106–120, 2019.

E. Blossom, “GNU Radio: Tools for Exploring the Radio Frequency Spectrum,” Linux Journal, vol. 2004, no. 122, 2004.

E. Lopelli, J. van der Tang, and A. van Roermund, “Architectural Trade-offs in Security,” in Proc. of 2017 Security and Privacy, 2017, pp. 90–102.