SnoopDog: Detecting USB Bus Sniffers Using Responsive EMR

Srinivasan Murali[†‡1,2], Youngtak Cho[†2], Huadi Zhu[†§1], Pan Li[¶], Ming Li[†]
[†] *The University of Texas at Arlington, Arlington, TX, USA*
[‡] *Texas A&M University-San Antonio, San Antonio, TX, USA*
[§] *Georgia State University, Atlanta, GA, USA*
[¶] *Case Western Reserve University, Cleveland, OH, USA*
Email: smurali@tamusa.edu, youngtak.cho@mavs.uta.edu, hzhu17@gsu.edu, lipan@case.edu, ming.li@uta.edu

Abstract—**The lack of encryption and authentication mechanisms in USB standards renders USB traffic susceptible to sniffing attacks. This paper presents an initial effort to detect USB bus sniffing through the development of a detection system, SnoopDog. It does not require hardware redesign of USB devices or modifications to the kernel or USB protocol stack. The system utilizes a *probe-and-detect strategy*. The host PC generates bait traffic with a dummy endpoint address. While benign devices discard this traffic due to the address mismatch, a sniffer captures the data, consequently emitting responsive electromagnetic radiation (EMR). To determine whether a USB device is a sniffer, SnoopDog calculates the correlation between the bait traffic and the responsive EMR signals captured near the target device. A high correlation indicates the presence of a sniffer. Recognizing that sniffer's EMR signals can be weak, we introduce a novel temporal folding scheme to improve the signal-to-noise ratio (SNR). To evaluate the performance, we build a prototype of SnoopDog and conduct comprehensive evaluations under a variety of settings, where SnoopDog delivers a promising detection accuracy with minor system overhead.**

1. Introduction

Background. Universal Serial Bus (USB) is the de-facto standard to connect computers with external peripherals, such as storage devices, network devices, audio devices, and hubs. According to a recent report, the USB devices market is expected to reach $59 billion by 2027 with a constant annual growth rate of 8.7% [81]. Over the past few decades, USB has evolved into multiple generations and versions and been adopted in a variety of devices and systems. At the same time, however, manufacturers of many end devices such as laptops have gradually reduced the number of available USB ports [78], in favor of slimmer and portable designs in recent years. Consequently, USB hubs have surged in popularity, allowing end-users to connect multiple USB devices of various types despite the limited number of ports [80].

[1] *This work was completed when the author was a Ph.D. student at The University of Texas at Arlington.*
[2] *Equal Contribution.*

Due to the *trust-by-default* nature, USB innovation has largely left security as an afterthought. Since USB devices must be physically attached to a host terminal to launch an attack, the USB Implementers Forum (USB-IF) has mentioned that "consumers should only grant trusted sources with access to their USB devices" [75]. It essentially delegates the end-user with the responsibility of restricting physical access rather than employing established security mechanisms such as encryption. This presents a security challenge since end-users often lack a comprehensive understanding of the technological risks involved [70]. Additionally, although USB requires physical access to launch attacks, prior studies have shown very cost-effective ways to motivate users to plug in USB devices to host terminals, such as branding USB drives with a government logo [54] and modifying their appearance as seemingly benign [33].

Motivation. As a result, there have been a multitude of USB attacks. Among them *off-path sniffing attacks* have been extensively exploited due to their low implementation overhead. For USB 1.x and 2.0 standards, downstream traffic is broadcast on the hub, making it observable for all devices on hub ports. It leaves room for sniffers to eavesdrop on all downstream communications carried over the same hub. ***Although the downstream traffic becomes unicast in USB 3.x and later versions, they are not immune from sniffing attacks.*** The vulnerability still exists when USB devices and hubs in different protocol versions are interconnected, due to the USB protocol's backward compatibility [19], [47].

To defend against such attacks, prior works [6], [47] suggested encrypting USB traffic. Oberg *et al.* [51] proposed using the unicast channel to replace the traditional broadcast channel on the USB bus. Recently, Dumitru *et al.* [18] built a USB Proxy prototype that acts as an intermediary between the host PC and USB devices, with multiple security functions implemented to resist USB bus sniffing. However, these approaches either require modifications of the kernel and USB protocol stack [6], [47], hardware redesign of USB devices [51], or are limited to supporting a narrow range of peripherals like mice and keyboards [18]. Additionally, while the existing strategies focus on proactively preventing sniffing attacks, we are interested in their detection, which has not been investigated before.

Our approach. In this paper, we explore a novel side channel to detect USB bus sniffers. Our key insight is that sniffers, like any other electronic devices, inevitably emit electromagnetic radiation (EMR) to the environment while being active, for example, when snooping data from the USB bus. We adopt a *probe-and-detect* strategy. The host PC generates some "bait" traffic following a certain pattern, where the packets are embedded with a dummy address. As the sniffer copies the bait traffic, it is expected to produce responsive EMR signals that mirror the bait pattern. Benign devices, on the other hand, do not exhibit such a pattern since they would discard a packet once noticing an address mismatch. By leveraging the distinctive pattern of the EMR signals produced in response to the bait traffic, we can effectively identify the presence of a USB sniffer.

Despite the promising idea, we are still faced with the following challenges. *Firstly*, intelligent sniffers may perform analysis of the traffic seen on the bus to recognize any suspicious bait traffic and then employ appropriate countermeasures to avoid being detected. *Secondly*, an attacker may apply metal casing or shielding to reduce EMR emanation. It may impact the detection accuracy consequently. *Finally*, as the hardware components across sniffers are diverse due to manufacturers' choices, so are the corresponding EMR frequency characteristics. Hence, the traditional spectrum analysis techniques for device fingerprinting are not suited in our case.

To tackle the first challenge, we perform statistical analysis to characterize the pattern of common USB traffic types. We propose to generate bait traffic mimicking common USB application traffic. Hence, the bait traffic will appear normal from the sniffer's perspective. To counteract the second challenge, we propose a novel temporal folding scheme to boost the signal-noise ratio (SNR) of the EMR signal by aggregating multiple EMR segments. Lastly, we observe that the sniffer's responsive EMR aligns well with bait traffic in timing. We thus compare the correlation of these two temporal sequences to decide if a target device is a sniffer.

Our contributions. In this paper, we develop a detection system called SnoopDog to monitor USB bus sniffers. It consists of two components: software at the host PC to generate bait traffic and a detector to capture and analyze EMR emitted from the target device. To evaluate its performance, we build a prototype, with the detector mainly composed of a software-defined radio (SDR), an MCU, and a near-field probe. We conduct evaluations on 20 hubs, 12 USB devices, and 4 sniffers with diverse sizes, configurations, and USB standard versions, from various manufacturers. The evaluations consider a variety of settings, such as attack configurations, and impact factors. We also fine-tune the system parameters of SnoopDog to explore its optimum performance as well as benchmark its system overhead. Our results show that SnoopDog's performance is robust across a variety of sniffers, hubs, USB devices, and attack configurations. It achieves a detection accuracy of 100% when the measurement distance is within 3 cm, which is necessary to pinpoint a sniffer among multiple USB devices.

The contributions of this work are summarized as follows:

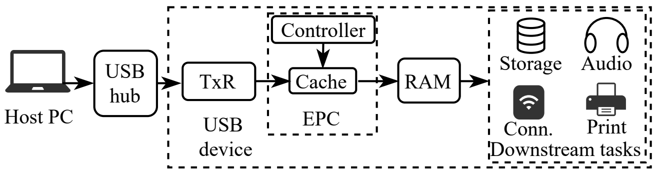


Figure 1: USB downstream communication data flow.

- This work presents the first attempt to detect USB sniffing attacks. We develop a detection system SnoopDog that works in a *plug-and-play* manner: It does not require hardware redesign of USB devices or modifications to the kernel or USB protocol stack.
- From the technical aspect, SnoopDog exploits time-domain patterns in the EMR to detect USB sniffers, instead of frequency-domain fingerprinting commonly used in existing EMR-based device recognition. Furthermore, we propose a novel temporal folding scheme to enhance EMR signal quality.
- We build a proof-of-concept prototype of SnoopDog. Comprehensive evaluations are conducted to evaluate its performance. It delivers promising detection accuracy.

2. Background

2.1. USB Downstream Communications

The USB communication process is established to allow a host to communicate with a connected USB device effectively. As a USB device is plugged into the USB port, the device and host undergo an *enumeration stage*, from which the host assigns a unique address to the device. After this, when the host intends to transfer files to the USB device, it first sends a *token* packet, indicating the type of transaction. Token packets are embedded with the intended USB device address. According to the USB standard, devices must only process and respond to tokens addressed to them while ignoring others. This is closely followed by the flow of *data* packets. This means that only the intended device would receive the data packets, while the others would ignore the data packets that are not addressed to them.

On the receiver side, the incoming data are first captured by the USB *transceiver* (TxR) and then transferred to the *endpoint controller* (EPC), where they are validated for the address and decoded. If the address embedded in the packet matches the device's assigned address, the packet is transferred to the next block in the pipeline, e.g., RAM; otherwise, the packet is discarded from the cache by the controller. The final destination of incoming packets may be a storage module, a network connector, an audio card, or a printer, depending on the respective downstream task [29], [73]. The entire process is depicted in Figure 1.

RAM, or an equivalent component, is not mandatory in simpler USB devices such as keyboards and mice. However, it is commonly found in more sophisticated USB devices that require rapid data processing, storage capabilities, or real-time task management. For a USB sniffer, RAM is crucial for effectively capturing data traffic across a wide range of data rates on the bus.

2.2. USB Downstream Communications Involving USB Hubs

To expand the number of USB ports available, USB hubs can be linked in a chain, with up to five hubs in sequence. This setup allows for the support of up to 127 devices in total.

Traffic forwarding characteristics of USB hubs. When the hub is in USB 1.x or 2.0 standard, the downstream traffic is *broadcast*, i.e., the traffic is forwarded to all the downstream ports of the hub [13]–[15]. Hence, although the traffic is meant for one specific device, all devices on the hub receive a copy of the traffic. However, only the intended recipient, whose address matches the traffic's encapsulated address, processes it further while the other devices discard the packet at the EPC module. For a hub in USB 3.x or later versions (e.g., USB4), the downstream traffic becomes *unicast* [27], [28], [74]. Upon receiving a packet from the host, the hub examines the address embedded in the packet header. Then the packet is forwarded to the intended downstream port. This is made possible by the dedicated super-speed pins/data lines present in hardware.

Form factors of USB hubs. USB hubs have both embedded and external forms. The former, or called USB bus, is integrated directly into a PC. They provide multiple USB ports while being part of the PC's internal hardware. The latter is a standalone device that connects to a single USB port on a host, offering additional ports externally. Despite these physical differences, the underlying functions and protocols of both forms are fundamentally the same as long as they adopt the same standard. It is worth mentioning the *root hub*, which distinguishes itself from the regular hubs discussed above. As an integral part of the host PC, the root hub operates in accordance with the Extensible Host Controller Interface for USB (xHCI) specification [31]. The presence of the root hub alters the nature of the downstream traffic (unicast or broadcast) based on the connected device's requested USB version.

2.3. Sniffing Attacks on USB Downstream Communications

For USB 1.x and 2.0 standards, downstream traffic is broadcast on the hub, making it observable for all devices on hub ports [19], [47]. It leaves room for sniffers to eavesdrop on all downstream communications carried over the same hub. Even worse, as USB traffic is not encrypted by default, the data is readily available to the sniffer in plaintext format. Such an attack, named *off-path sniffing attack*, was first identified by Neugschwandtner *et al.* [47] and later widely recognized by the community [6], [19], [37], [48], [63], [69].

Although later versions (e.g., USB 3.x and USB4) adopt a unicast downstream traffic model, there are still numerous scenarios susceptible to sniffing attacks. This happens when USB devices/hubs in different protocol versions are interconnected. For instance, if a USB 2.0 flash drive is connected to a USB 3.x hub, then the downstream traffic to the flash drive would still be broadcast [19], [47]. This is because USB 3.x hubs incorporate USB 2.0 data lanes in parallel with USB 3.x hardware to ensure backward compatibility. When a USB 2.0 device is connected to a USB 3.x hub, the USB 2.0 data lanes are activated to communicate with the device. Hence, the downstream traffic becomes broadcast even though it is carried through a USB 3.x hub. In Appendix A, we provide a more thorough investigation regarding the scenarios susceptible to sniffing attacks. Generally, a victim device is vulnerable to sniffing attacks as long as any entity, including the hubs and the victim itself, in the downstream traffic chain adopts USB 2.0 or earlier. Here, a "chain" starts from the host PC and ends at the victim device. In summary, ***USB sniffing will remain a critical problem for many years due to the coexistence of USB devices of different protocol versions in daily life***.

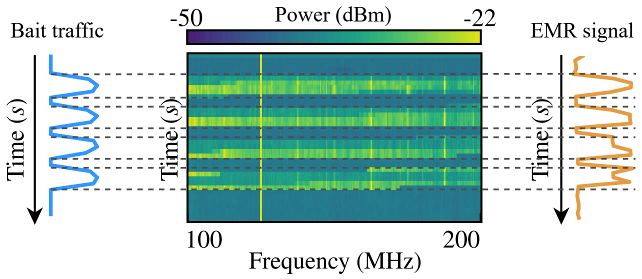
It is worth mentioning the *on-path sniffing attack* [17], [38], where the sniffer resides in between the host and the victim device to listen to traffic in both ways. This attack typically requires modifications to a USB port or a hub, rendering it much less practical to launch than the *off-path sniffing attack* that simply plugs a malicious USB device into a hub port. While this work primarily concentrates on off-path sniffing detection, SnoopDog is readily applicable to on-path sniffing. This is because SnoopDog monitors the EMR response to bait traffic for sniffer detection, which is irrespective of sniffer's attack path.

2.4. Threat Model

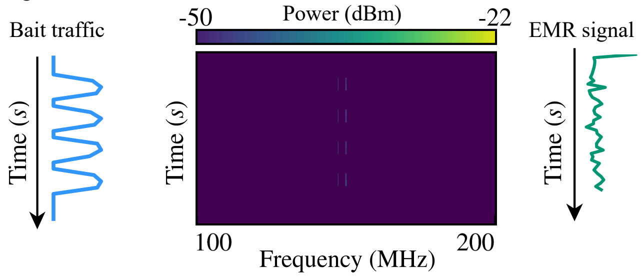
Attack model. We consider off-path sniffing attacks* described in §2.3; the victim device's downstream traffic is broadcast in the USB bus to which the attacker has physical access. The sniffer's primary goal is to copy downstream traffic seen on the bus. The sniffer can perform basic traffic pattern analysis to recognize any suspicious bait traffic. It may also adopt strategic sniffing to avoid being detected. For instance, rather than copying all data packets from the bus, it might perform selective or random eavesdropping to reduce its detectability. Meanwhile, for a sniffer to carry out a successful attack, it should remain active for a meaningful time duration [47]. The detection of sniffers that are entirely in an inactive state is out of the scope of this work.

The infiltration vector for the sniffer into the USB hub takes multiple forms. One feasible method is to integrate the sniffer module within normal-looking USB devices, such as USB lamps, fans, laptop cooling pads, and flash drives [63]. These devices are often produced by manufacturers without brand recognition. Users may unknowingly purchase them from online markets [8], [50], [72], [82], pick them up in public spaces, or receive them as freebies [12], [76].

$*$. For the rest of the paper, we use the term sniffing attack to refer to off-path sniffing attack for expression simplicity without causing confusion.



(a) Characteristics of sniffer's EMR signals. Left: Bait traffic. Middle: Spectrogram of EMR signals. Right: Time-domain EMR signals.



(b) Characteristics of benign device's EMR signals. Left: Bait traffic. Middle: Spectrogram of EMR signals. Right: Time-domain EMR signals.

Figure 2: The result demonstrates that the sniffer's EMR correlates with the bait traffic in timing. On the other hand, such a phenomenon is not observed for the benign device.

Detection model. SnoopDog aims to identify USB sniffers in diverse and complex environments. It has no prior knowledge about the sniffer's type or hardware specifications. The detector is implemented as a portable device so that users can move around while scanning for EMR emanations from target devices. As the detector is built with a near-field probe, a user needs to place the probe close to each target device to pick up meaningful EMR measures for detection. SnoopDog is designed as a user-initiated detection tool. It allows users to perform detection at their discretion, for example, when a new USB device is connected to the bus or once in a while as needed. In practice, the USB bus may be shared by other ongoing USB services, such as file transfer, video/audio streaming, and command communications with peripheral devices. SnoopDog is designed to operate concurrently with these USB services.

3. Preliminaries of EMR from Sniffers

This section studies the characteristics of sniffer's EMR. Additionally, it identifies the specific circuit modules generating EMR. The results motivate the design of SnoopDog.

3.1. Characteristics of EMR

Experimental setting. To demonstrate the characteristics of EMR signals from the sniffer, we consider a basic experimental setup where a sniffer and a benign USB device are connected to a USB hub that is further connected to a host PC (Figure 14 in Appendix B). Following prior work [47], we employ the Beagle USB 480 protocol analyzer [71] as the sniffer. USB protocol analyzers are commonly used as sniffers due to their capabilities for real-time packet capturing and decoding directly from the USB bus. These functions mirror those typically found in a USB sniffer. The host is programmed to generate bait downstream traffic, where the packets are embedded with a dummy endpoint address. In other words, the bait traffic is addressed to a USB device that is not physically present. The EMR signal collection and analysis are performed using a Tektronix MDO34 oscilloscope.

Results. Figure 2a exhibits the characteristics of sniffer's EMR signals under bait traffic. It is observed that these two align well in timing; that is, the EMR real-time amplitude positively correlates with the instant rate of bait traffic. This is caused by sniffer's behavior in copying bait packets from the USB bus. As the circuit is active (i.e., copying data), it emits EMR to the environment. As a comparison, we also show in Figure 2b the characteristics of EMR signals of the benign device under the same bait traffic. The measured EMR is much weaker without any recognizable pattern related to the bait traffic. This is because the benign device simply discards a bait packet once noticing an unmatched address. We are thus motivated to leverage the distinctive pattern in EMR between the sniffer and benign device to decide if a target device is a sniffer.

3.2. Sources of EMR

When a sniffer is active, one of the prominent carrier frequencies is its RAM's operational frequency f_c. As discussed in §2.1, the sniffer continuously captures and writes data from the USB bus to its RAM, even those not intended for it. Meanwhile, the non-ideal and non-linear circuits in the sniffer generate harmonics at the $n \times f_c$ frequencies, where n is an integer. As shown in Figure 15a and 15b in Appendix B, the fundamental carrier frequency f_c is 187.5 MHz, which matches the RAM's programmed operational frequency of the sniffer (i.e., Beagle USB 480 protocol analyzer) [30]. In addition, its harmonics are found at 375 MHz, 562.5 MHz, $\cdots$, following the expression of $n \times f_c$.

Aside from RAM, a sniffer is typically composed of TxR, EPC, and other peripheral hardware (Figure 1), which also generate carrier frequencies. In the expeirment, we detect the following frequency components: 24 MHz, 60 MHz, and 125 MHz, which correspond to the clock frequencies of the external on-board oscillator, TxR internal reference clock, and EPC, respectively. (A more detailed frequency analysis is given in Appendix B.) The sources are confirmed by comparing experimental measures with the numbers provided in the related datasheets [20], [30], [43], [61].

Due to the small size of USB devices (including sniffers), several circuit modules reside on the same PCB substrate in practice. As a result, it gives rise to the effect of *carrier inter-modulation & coupling* [11], [42]. It generates

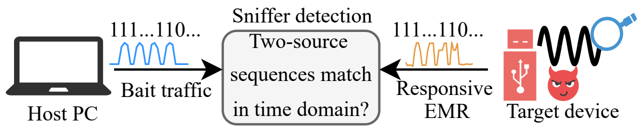


Figure 3: To determine whether a target USB device is a sniffer, SnoopDog examines the correlation between two temporal sequences: the bait traffic generated by the host PC and the responsive EMR signals captured near the target device. A high correlation between these two sequences indicates the presence of a sniffer.

additional signals or frequencies. This is represented as $f = n_1 \cdot f_{c1} + n_2 \cdot f_{c2} + \cdots + n_k \cdot f_{ck}$, where f_{ck} represents the carrier frequency from the kth circuit module, and n_k is its integer coefficient. Consequently, ***the sniffer's EMR spreads across a wide frequency band from MHz to GHz***. In the meantime, the signal is continuously attenuated at higher frequencies.

Discussions. Our analysis reveals that EMR signals are emitted from various circuit modules within the sniffer, compounded by circuit non-linearity and inter-modulation & coupling effects. As a result, sniffer's EMR spectral components are fairly complex, consisting of fundamental carrier frequencies, their harmonics, and even their combinations. This complexity renders their characterization challenging. There are existing works utilizing frequency-domain characteristics of EMR for device recognition (e.g., [9], [55], [59], [84]). They assume that the target device's EMR frequency composites or patterns are known *a priori*. Apparently, these approaches are not suited here. Fortunately, as demonstrated in §3.1, sniffers exhibit unique patterns in their EMR in the time domain. Therefore, ***we propose to explore the temporal characteristics, instead of the spectral ones, in the EMR for sniffer detection***.

4. SnoopDog Design

Overview. SnoopDog consists of two main components: software on the host PC and a front-end detector. The software is designed to send downstream bait traffic to the USB bus in a specific pattern, embedding these packets with a dummy endpoint address. Benign USB devices will discard these packets upon detecting the address mismatch in their cache. Conversely, a sniffer copies all traffic to its RAM, thereby generating responsive EMR signals that closely mirror the bait traffic pattern, as analyzed in §3.1. To determine whether a target USB device is a sniffer, we examine the correlation between two temporal sequences: the bait traffic generated by the host PC and the responsive EMR signals captured near the target device. A high correlation between these two sequences indicates the presence of a sniffer. The functions of EMR measurement, signal processing, and temporal sequence comparison are all integrated into the detector; its prototyping details will be presented in §5. The detector is connected to the host via a USB port, which facilitates communication between the two components, specifically, transmitting the bait traffic pattern to the detector for temporal sequence analysis. The high-level design idea of SnoopDog is presented in Figure 3.

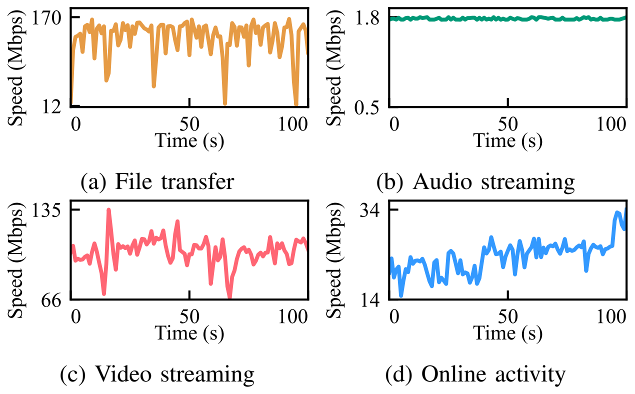


Figure 4: Traffic patterns of four common USB applications.

In the following, we elaborate on the design details of the host PC and detector in §4.1 and §4.2, respectively. Realizing sniffers may apply *shielding* to reduce EMR emanations, §4.3 further develops a novel temporal folding scheme to enhance EMR signals.

4.1. Bait Traffic Design at the Host PC

One of the primary design considerations of the bait traffic is that it should appear "normal" to the sniffer; that is, it should look like regular USB traffic as much as possible. This is because intelligent sniffers may perform traffic analysis and employ countermeasures when necessary. To avoid being recognized as purposely designed traffic, we propose to create the bait traffic in a way that mimics commonly seen traffic on a USB bus.

We first perform a series of experiments to characterize the pattern of USB traffic generated by various applications. Four common USB applications are considered: *file transfer*, *audio streaming*, *video streaming*, and *online activity*. Note that the bait traffic is not restricted to these four types but can be of arbitrary format and transfer mode theoretically. File transfer traffic is generated by initiating file copy operations from the host to the connected flash drive. For audio streaming, the host streams music to an external speaker via a USB port. For video streaming, the host is connected to an external monitor through a USB-to-VGA adapter [62]. For online activity, the host uses a USB network adapter to continuously sync a file with a cloud platform. Figure 4 shows traffic instances of all four applications. Their patterns are diverse. For example, file transfer and video streaming exhibit relatively significant variations in data rates, whereas audio streaming tends to be stable. To better characterize these patterns, we perform a statistical analysis of key transmission metrics. The analysis provides valuable guidance for selecting appropriate parameters for the bait traffic. Specifically, we examine metrics such as *packets per microframe*, *packet size*, *queue time*, *service latency*, and *throughput*. Their definitions and analysis are given in Appendix C. We propose to pick the bait traffic parameters from their feasible range. (See details in Table 8.) The parameter settings remain fixed within a single detection round. They can vary across multiple detection instances to better reflect traffic dynamics in real-world scenarios.

4.2. Design at the Detector

Signal acquisition bandwidth consideration. According to the analysis (§3.2 and Appendix B), sniffer EMR spreads across a wide spectrum from MHz to GHz due to circuit non-linearity and coupling effects. We thus propose to collect EMR from any sub-band of reasonable bandwidth (e.g., a couple of hundred MHz) within this range. This approach is different from existing works on EMR-based device recognition (e.g., [9], [55], [59], [84]), where receivers have to tune to specific frequencies for signal pickup. Such methods utilize EMR frequency-domain characteristics for device fingerprinting. Hence, they require the target device's EMR frequency composites or patterns to be known *a priori*, unlike our scheme.

The implementation of the front-end detector utilizes a software-defined radio (SDR) to sweep across the target frequency band for signal acquisition. A key design consideration is to balance signal acquisition efficiency and accuracy: a wider scanning range captures more of the sniffer's spectral components but increases scanning time, while a narrower range improves efficiency at the potential cost of missing some components. The specific parameter settings will be presented in §5.

Denoising. Upon the capturing of raw EMR signals, it is crucial to eliminate unwanted noise via denoising. Generally, this noise originates from two primary sources: nearby electronic devices (e.g., trackpads, keyboards, speakers, the host PC, and smartphones) and circuit modules within the sniffer itself (e.g., TxR, EPC, and oscillators). To remove them, we first apply the fast Fourier transform (FFT) to convert the time-series EMR raw readings into its frequency-domain representation. Then a bandstop filter is employed to remove the noise at specific frequencies. This idea is faced with a challenge: As the detector has no prior knowledge of the background environment or the sniffer, it is unaware of the frequencies where the unwanted noises reside. To overcome this, we propose to perform a baseline measurement of the noise. Specifically, we measure the EMR around the target device when it is in standby mode (i.e., without bait traffic). The measurement then captures the noise. Through spectrum analysis, we can easily derive its frequency components. The baseline measurement is performed for each new detection instance to cope with dynamic environments.

Sniffer detection. The idea is to compare the correlation of two-source temporal sequences: the processed EMR signal and bait traffic. We first extract the former. Specifically, the time horizon is divided into fixed-length time interval Δt as shown in Figure 6. If its instant value within Δt is larger than a threshold ρ, it is labeled as 1; otherwise, it is labeled as 0. The derived time-series binary sequence is denoted as S_e, where the subscript e stands for EMR. We empirically set Δt to 100 ms. The remaining task is to set a proper value ρ. To this end, we propose to perform a baseline measurement of the background noise when there is no active bait traffic. Specifically, we calculate its average μ and standard deviation σ. Then, we set $\rho = \mu + 2\sigma$, 2 standard deviations above the average background noise baseline. Next, we adopt a similar approach to extract the other binary sequence S_b from the bait traffic. Likewise, we divide the bait traffic into equal time intervals Δt. Then, we assign 1 to an interval if its instant data rate is above a threshold and 0 otherwise.

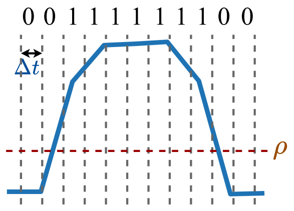


Figure 6: Generating temporal sequence S_e from the processed EMR.

Lastly, by comparing the two sequences S_e and S_b, SnoopDog verifies the presence of a sniffer. If both sequences S_e and S_b match well, the target device is detected as a sniffer; otherwise, it is not. To this end, the classic dynamic time warping (DTW) is applied. Once the detection is positive, SnoopDog triggers a warning to the end-user for their appropriate further action. There are several classic approaches to quantify the correlation between two sequences, including *Pearson correlation*, *Minkowski distance*, and *cosine similarity*. Through testing, DTW outperforms the other three in our case. The evaluation details will be presented in §5.4.

Detection of strategic sniffers. Sniffers may adopt strategic tactics when launching attacks. For example, rather than copying all data packets from the bus, they may selectively copy traffic fragments or perform sporadic eavesdropping. This part discusses how to deal with strategic sniffers. First of all, as long as a sniffer copies data, it inevitably emits unintentional EMR within the transmission duration of the bait traffic, a behavior not present in benign devices. However, the responsive EMR measured near the strategic sniffer would not follow the bait traffic in timing perfectly. Luckily, DTW bears the necessary flexibility in quantifying the correlation between two sequences even though they are not perfectly aligned. Nonetheless, timing misalignment can reduce the accuracy of correlation calculations, and thus impact detection. To mitigate this, we propose adopting an extended observation window to collect additional EMR samples for decision-making. The increased sample size enhances the detector's ability to identify sniffing activities that shorter windows might miss. The detection performance against strategic sniffers will be discussed in §5.2.

4.3. EMR Enhancement via Temporal Folding

§4.1 and §4.2 present the basic modules of SnoopDog. In practice, the sniffer's circuit is concealed inside an enclosure. Some of them may even apply *shielding* to reduce the EMR emanations. Hence, EMR from sniffers can be weak. To address this issue, we further develop a novel

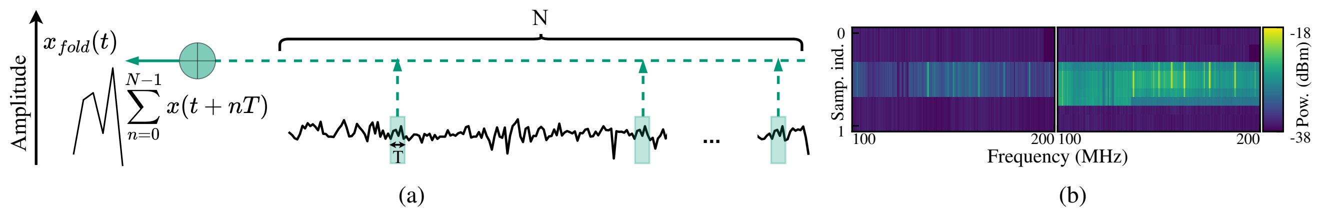


Figure 5: (a) Illustration of the proposed temporal folding. Since the detector knows where the intentionally embedded segments are, it can selectively extract these segments from the received signal and aggregate them in the time domain. (b) Folding effect. Left: Before folding. Right: After folding.

signal folding scheme to enhance the EMR signal quality at the detector.

The topic of enhancing the signal strength of a radiating target has been studied for a while. To pick up weak signals from background noise, one classic approach is to group and sum the energy of small signal components in the frequency domain, i.e., harmonic frequencies that periodically appear across a wide spectrum. This technique is called *spectrum folding* and adopted by recent works on EM/RF-based device detection. However, this technique is unsuited here as sniffer's EMR spectral components are fairly complex (§3.2). Recognizing the harmonic frequencies is challenging in the first place. Instead, we propose to "fold" the signal in the time domain.

To facilitate the proposed temporal folding, we slightly modify the bait traffic by inserting some repeated "segments" at random times during the bait traffic stream. A segment is a small burst of traffic that appears like the rest part of the bait traffic. As the segments only take a small portion (e.g., 1%) of the entire bait traffic, it is barely noticeable by the sniffer.

Denote the segment duration as T. Then the responsive EMR from the sniffer should include signal segments of the same duration T. Since the detector knows where these segments are, it can selectively extract these segments from the received signal. Our proposed temporal folding is to aggregate N EMR segments by adding them together: $x_{fold}(t) = \sum_{n=0}^{N-1} x(t+nT)$, where $x(t)$ is the instant EMR segment measurement at time t, $x_{fold}(t)$ is the aggregated signal via temporal folding, N is the number of segments over which the signal is folded. Figure 5a illustrates our idea. Figure 5b further shows its effect. Apparently, the EMR signal strength is greatly enhanced via our folding scheme, whereas the background noise is not. This is because the noise, being random and uncorrelated from one segment to another, tends to cancel each other, while the EMR segment measures, being well aligned, are added up by aggregating over multiple of them.

5. Prototype Implementation and Evaluation

The implementation effort of SnoopDog consists of two parts: building the hardware detector and developing software at the host PC.

Detector prototyping. To facilitate the acquisition of EMR signals near the target device, we build a detector consisting of a near-field probe [64], an RTL-SDR software-defined radio device [3], an RPi 4B [56], and a battery [2], as shown in Figure 7. The RTL-SDR is able to receive EMR signals at the frequency range of 24 MHz to 1766 MHz, which overlaps the spectrum of interest here, i.e., from MHz to GHz. Its antenna pin is connected with a Tekbox NF probe [64]. In our implementation, the RTL-SDR is set to sweep across the frequency spectrum between 100-200 MHz for signal acquisition to balance acquisition efficiency and accuracy. Note that the scanning frequency range does not have to fix at 100–200 MHz. The sniffer EMR is detectable at any sub-band of a couple of hundred MHz within the MHz-GHz frequency range according to the analysis. After captured by the probe, the EMR signals are digitized by the SDR and transmitted to the RPi for further processing and analysis. The RPi serves as the detector's MCU, where the detection decision is made. This prototype has a compact dimension of 16 cm $\times$ 6 cm, rendering it portable for users to perform detection.

Software development. We utilize Dell Precision 3260 tower as the host PC, which is equipped with an Intel Core i9 processor and 32GB RAM, running Ubuntu 20 LTS. The host PC is installed with a Python script to manage the bait traffic generation. The bait traffic is designed to mimic normal USB file transfer, following the statistical analysis in Appendix C. To enable the host PC to transmit the bait traffic pattern to the detector for temporal sequence analysis, these two are connected via a USB port.

5.1. Experimental Setup and Devices

Experimental setting. The basic experimental setup for evaluation is shown in Figure 7. The host PC is extended with a USB hub that is further connected to USB device(s) and a sniffer. Our goal is to evaluate the performance of SnoopDog in recognizing benign USB devices and sniffers under various settings and environmental conditions. Unless otherwise specified, the results are obtained under optimal system parameter settings, with the probe positioned directly adjacent to the target device and without shielding applied to the sniffer.

USB devices, hubs, and sniffers. A total of 20 USB hubs are selected for evaluation, taking into account various factors such as supported USB versions, port types, brands, and form factors. Additionally, we test 12 USB devices, including flash drives, network adapters, and sound adapters.

TABLE 1: Detection performance against different sniffers.

#	Metric	TPR(%)	TNR(%)	Acc.(%)	Pre.(%)	SNR(dB)
1	Cynthion	100	100	100	100	5.10
2	Beagle	100	100	100	100	5.51
3	Mercury	100	100	100	100	5.45
4	OpenVizsla	100	100	100	100	2.53

TABLE 2: Detection performance in different connection configurations discussed in Appendix A.

	Case 1 (Figure 11)				Case 2 (Figure 12)							
Conf.	1	2	3	4	1	2	3	4	5	6	7	8
TPR(%)	100	100	100	-	100	100	100	100	100	100	100	-
TNR(%)	100	100	100	-	100	100	100	100	100	100	100	-

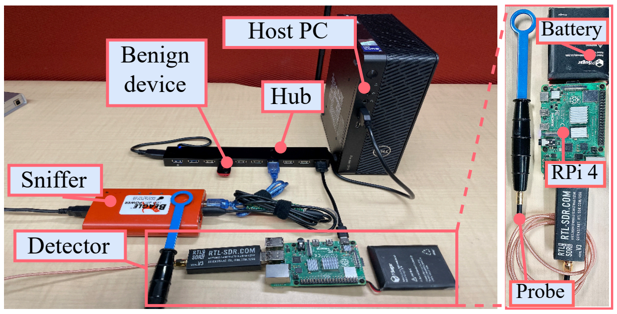


Figure 7: Left: Experimental setup for evaluation. Right: Detector prototyping.

Four different sniffers are evaluated: Cynthion [21], the Beagle 480 USB protocol analyzer [71], Openvizsla [52], and the Mercury T2 analyzer [5]. Specifically, Cynthion is an open-source USB sniffer [22], [23]. The other three are USB protocol analyzers. Following prior work [47], we also employ analyzers as sniffers because they can capture and decode packets from the USB bus in real time. These capabilities closely resemble the functionality expected of a USB sniffer. Their pictures are given in Figure 16 in the appendix.

Data collection and evaluation metrics. To comprehensively evaluate the efficacy and usability of SnoopDog, we consider a broad spectrum of factors including device heterogeneity, connection settings, and measurement conditions. We conduct 25 trials for each configuration. The following metrics are adopted. (a) *True positive rate (TPR)* denotes the probability that SnoopDog correctly identifies a sniffer; (b) *True negative rate (TNR)* denotes the probability that SnoopDog correctly identifies a benign USB device; (c) *Accuracy* measures overall detection performance: $\frac{TP+TN}{\sum \text{samples}}$; (d) *Precision* indicates the quality of SnoopDog's positive detection results: $\frac{TP}{TP+FP}$.

5.2. Overall Performance

We first present the overall performance of SnoopDog across different sniffers, hubs, and attack configurations.

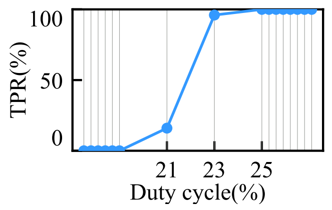
Different sniffers. We evaluate SnoopDog against four sniffers mentioned above. The result is given in Table 1. We find that SnoopDog delivers consistent detection accuracy of 100% across the four sniffers, while the SNR of their measured EMR signals varies. It suggests that SnoopDog can be used to detect sniffers with different form factors and circuits. Note that SnoopDog does not require prior knowledge regarding sniffer's EMR spectral characteristics or frequency components. This is because SnoopDog utilizes the temporal patterns in sniffer's EMR for the detection.

TABLE 3: List of USB hubs tested and the detection performance. ●: Hubs in USB 2.0; ○: Hubs in USB 3.x; ◐: Hubs with both 2.0 and 3.x ports.

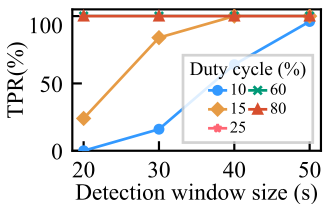
#	Hub name	Ports	Acc. (%)	TPR (%)	TNR (%)
1	Acer USB ODK350	●	100	100	100
2	Acer USB-C ODK360	○	100	100	100
3	AmazonBasics U3-10HUB	◐	100	100	100
4	AmazonBasics HU2W70E1	●	100	100	100
5	Anker 332	○	100	100	100
6	Anker A7515	○	100	100	100
7	Belkin F4U042BT	●	100	100	100
8	HitPromo USB	●	100	100	100
9	j5create JUH377	○	100	100	100
10	novoo NH07D-111R	○	100	100	100
11	Plugable USBC-7IN1	○	100	100	100
12	SABRENT HB-UM43	●	100	100	100
13	SABRENT HB-MCRM	●	100	100	100
14	StarTech.com ST4200MINI	●	100	100	100
15	StarTech.com ST4200USB	●	100	100	100
16	Targus ACH114US	◐	100	100	100
17	TrippLite U223-004-IND	●	100	100	100
18	UGREEN USB C-(30758)	○	100	100	100
19	Unnamed	●	100	100	100
20	Yoigo USB hub	●	100	100	100

Different connection configurations. USB devices are susceptible to sniffing attacks not only under USB 1.x and 2.0, but also in later versions when they coexist. This part evaluates the performance of SnoopDog under different connection configurations, specifically, the ones investigated in Appendix A. The results are shown in Table 2. In case study 1, the host PC is extended with one USB hub, which is connected to a victim device and a sniffer, shown in Figure 11. In case study 2, two hubs are concatenated in sequence; the victim device and the sniffer are connected to the same hub, shown in Figure 12. The result is promising: SnoopDog delivers 100% TPR and TNR across all vulnerable configurations.

Different hubs. We further examine whether the hub would impact the detection accuracy. For this purpose, a total of 20 hubs are tested. They are from a variety of manufacturers. Some of them work purely under 2.0 or 3.x standard, while some others support both kinds of ports. To ensure the setup with 3.x hubs is vulnerable to sniffing attacks, we adopt the benign devices in 2.0. The result is provided in Table 3. We find that SnoopDog delivers a consistent detection accuracy of 100% across all hubs.



(a) Impact of duty cycle.



(b) Impact of detect. win. size.

Figure 8: Performance against strategic sniffers.

Performance against strategic sniffers. Sniffers may be strategic to randomly copy a portion of traffic from the bus. In the experiment, we adopt the concept *duty cycle* to model the sniffer's behavior. It represents the percentage of time the sniffer is active during the total observation window. The sniffer randomly switches on (i.e., active) and off (i.e., inactive). A 100% duty cycle indicates that the sniffer copies all the data on the bus, while a 0% cycle means no sniffing occurs. We vary the duty cycle from 10% to 80%. Figure 8a presents the results. SnoopDog can effectively detect strategic sniffers with an active duty cycle of 25% and above. However, its effectiveness diminishes, when the active duty cycle becomes less. Meanwhile, if the strategic sniffer's active duty cycle drops below 25%, it collects very limited data from the bus and thus poses limited threat to the system. The above result is obtained when the observation window is set to 20 s.

We further examine the impact of the observation window size on detection performance against strategic sniffers in Figure 8b. Our results indicate that more "aggressive" sniffers, which adopt a larger duty cycle, require a smaller effective detection window for accurate identification. For instance, with a duty cycle of 25% or higher, SnoopDog achieves a 100% TPR using a window size of just 20 s. When the duty cycle decreases to 15%, the window size doubles to 40 s to maintain a 100% TPR. These findings are promising: adjusting the detection window size proves to be an effective strategy to counter strategic sniffers. Besides, a window size as short as 20 s allows SnoopDog to detect all sniffers with duty cycles of 25% and above, covering the majority of meaningful sniffer activities.

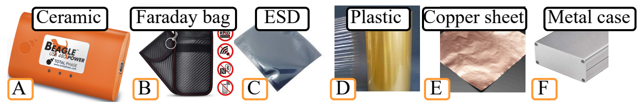
5.3. Comprehensive Evaluation of SnoopDog

In this section, we evaluate the performance of SnoopDog under different experiment settings.

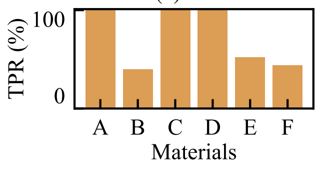
Co-existence with other USB devices. In practice, a sniffer may co-exist with other (benign) USB devices connected to the same hub. This experiment aims to determine whether benign devices produce interfering EMR signals that could potentially affect detection accuracy. We test 12 USB devices, including flash drives, network adapters, and sound adapters. The results, listed in Table 4, show that performance remains consistent despite the presence of various USB devices. Benign devices discard bait packets at their EPC upon detecting an address mismatch. In contrast, sniffers continue to write the data to RAM, emitting EMR that follows the bait traffic pattern. Therefore, the EMR patterns emitted by benign devices are distinctly different from those of sniffers.

TABLE 4: Impact of co-existence with other USB devices. 📶: Network; 📄: Drive; 🎧: Audio

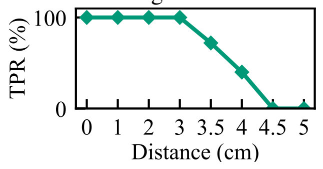
#	Device name	Type	Version	TPR(%)	TNR(%)
1	AmazonBasics Ultra	📄	3.x	100	100
2	BrosTrend WiFi	📶	3.x	100	100
3	Kexin	📄	2.0	100	100
4	Kingston DataTraveller	📄	3.x	100	100
5	Lexar S47	📄	3.x	100	100
6	Lexar V40	📄	2.0	100	100
7	PNY Attaché 4	📄	2.0	100	100
8	SanDisk Cruzer Blade	📄	2.0	100	100
9	SanDisk Ultra Luxe	📄	3.x	100	100
10	StarTech.com	🎧	2.0	100	100
11	Transcend TS64GJF730	📄	3.x	100	100
12	Verbatim ToughMAX	📄	2.0	100	100



(a) Different materials for testing.



(b) Impact of enclosure materials.



(c) Impact of measurement distance.

Figure 9: Impact of measurement settings.

Casing and shielding. A crafty sniffer can be concealed within an enclosure. Metal materials may also be applied to shield EMR signals. Thus, we evaluate the performance of SnoopDog using various casing materials, including a ceramic case, Faraday bag, electrostatic (ESD) bag, plastic bag, copper sheet, and metal case, as depicted in Figure 9a. The result is presented in Figure 9b. SnoopDog's TPR remains stable at 100% with ceramic, ESD, and plastic casing materials. However, TPR experiences a noticeable drop when the sniffer is enclosed within a Faraday bag (40%), wrapped with a copper sheet (52%), or concealed by a metal case (44%). ***Complete blockage of EMR by metal casing, in our case, is unlikely in practice. Given that the sniffer must connect to the USB bus via its connector, the interfacing renders EMR leakage inevitable.***

Measurement distance. We investigate the effect of varying the measurement distance between the probe and the sniffer, from 0 cm to 5 cm. As depicted in Figure 9c, increasing the distance leads to a decline in performance. Specifically, detection accuracy remains at 100% when the distance is within 3 cm, but it falls to 70% at 3.5 cm and further drops to 40% at 4 cm. This is because our detector employs the near-field probe to capture EMR signals. It is only effective at short distances. Users are thus suggested to position the probe close to the target device during detection.

TABLE 5: Impact of concurrent USB applications.

Application	File	Audio	Video	Online
TPR(%)	88	100	92	100
TNR(%)	100	100	100	100

A long detection distance is not necessarily preferred in our case. Our objective is not only to detect but also to precisely pinpoint the sniffer among multiple USB devices. A long detection distance would undermine this goal.

Concurrent USB applications. In practice, the USB bus may be shared by other ongoing USB applications. We now examine the detection performance when there is another active USB service. Four different USB applications are considered: file transfer, audio, video streaming, and online activity. As shown in Table 5, SnoopDog achieves 100% of TPR and TNR with concurrent online activity and audio service. However, TPR degrades slightly to 92% and 88% for video streaming and file transfers respectively. This may be attributed to the occasional high traffic bursts for these two applications (as shown in Figure 4). They introduce corresponding EMR noise that impacts the sniffer detection.

5.4. Micro Benchmarks

Bait traffic data rate. Figure 10a depicts the relation between the bait traffic rate and the EMR measured at the sniffer. We observe a clear positive correlation between these two: A higher traffic rate leads to a stronger EMR signal. This is because the former causes an intense circuit activity involved in data reception. The increased EM fields then result in stronger EMR emanation. We further examine its impact on SnoopDog's performance. Figure 10b shows that SnoopDog reaches 100% detection accuracy when the data rate is beyond 19 Mbps.

Bait traffic duration. We tune the bait traffic duration and examine how it influences the detection accuracy. Figure 10c displays the relationship between these two. Specifically, when the duration is 10 s, the TPR is 0%; it goes to 50% when the duration is 15 s and reaches 100% when the duration is 20 s. This observation meets our expectation: A longer bait traffic duration indicates more EMR segments to aggregate during temporal folding and more samples for a decision.

TABLE 6: Impact of folding to EMR SNR.

Distance (cm)	1	2	3	4
Without folding (dB)	2.80	1.97	1.73	-0.47
With folding (dB)	5.05	4.53	4.34	2.08

Effect of temporal folding. Recall that we develop temporal folding to enhance the weak EMR signal, as sniffers may be concealed in an enclosure or apply shielding. We obtain the SNR with or without folding by varying the measurement distance. Table 6 demonstrates that SNR experiences significant improvement via folding. For example, SNR is 1.97 dB without folding at a distance of 2 cm, whereas it becomes 4.53 dB with folding. Aggregating EMR in multiple segments effectively cancels the background noise and thus increases the SNR.

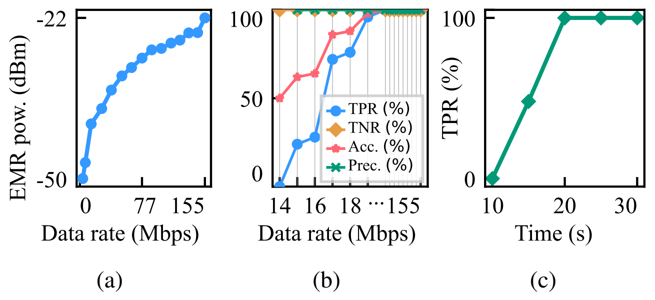


Figure 10: Micro benchmarks assessment. (a) EMR power strength vs. bait traffic data rate. (b) Impact of bait traffic data rate. (c) Impact of bait traffic duration.

TABLE 7: Comparison among different correlation calculation methods.

Metric/Method		Pearson correlation	Minkowski Distance	Cosine Similarity	DTW
Accuracy	Mean (%)	50	44	46	96
	Median (%)	88	88	82	98
	Std. Dev.	15.793	14.85	15.482	1.458

Selection of sequence correlation calculation method. Recall that SnoopDog determines whether to flag a device by comparing the correlation between two time sequences: one generated from bait traffic and the other from the measured EMR. To accomplish this, several classic methods have been considered, namely, DTW, Pearson correlation, Minkowski distance, and cosine similarity. Their performance is shown in Table 7. DTW outperforms the other three with the mean and median accuracy at 96% and 98%, respectively, due to its ability to handle temporal distortions and misalignment between the sequences compared. We consciously avoid employing any machine learning or AI methodologies for sniffing detection, due to their substantial overhead involved in data collection and model training.

6. Related Works

6.1. USB Sniffing Attacks and Defenses

USB sniffing attacks. The absence of encryption and authentication mechanisms in USB standards leaves USB traffic vulnerable to sniffing attacks. Depending on where the sniffer resides, the attack can be classified into *on-path sniffing* and *off-path sniffing*. Particularly, on-path sniffers are those sitting on the connection chain between the host and the victim device and eavesdrop on traffic [35], [36]. This type of attack typically requires modifications to a USB port or a hub. In contrast, off-path attacks do not have such restrictions. The adversary simply plugs a sniffer into a hub port. It then passively snoops on downstream traffic on the bus without modifying any component. As downstream traffic is broadcast on the hub for the USB 2.0 standard, it is thus observable for all devices, including sniffers, on hub ports. This attack vector was first reported by Neugschwandtner *et al.* [47]. Then pointed out in [19], [47], this vulnerability is not gone with the adoption of USB 3.x and later versions due to the USB protocol backward compatibility. Recently, the literature [63] reveals that upstream transmissions are susceptible to off-path sniffing as well by exploiting leaked signals in adjacent USB ports.

Defense against USB sniffing. Regarding defense, Neugschwandtner *et al.* [47] suggested encrypting USB traffic to protect against sniffing attacks. Similarly, Cinch [6] introduces a cryptographic overlay protocol to encapsulate all USB traffic via TLS sessions, utilizing a physical device called *crypto adapter*. Oberg *et al.* [51] proposed using deterministic time slots for each device to replace the traditional broadcast channel on the USB bus. More recently, Dumitru *et al.* [18] built a USB Proxy prototype that acts as an intermediary between the host PC and USB devices, with multiple security functions implemented. It is reported effective for resisting USB bus sniffing. However, these approaches either require hardware redesign of USB devices [51], modifications of the kernel and USB protocol stack [6], [47], or are limited to supporting a narrow range of peripherals like mice and keyboards [18]. In contrast, SnoopDog does not have such limitations. Moreover, while the existing strategies focus on preventing sniffing attacks, we are interested in their detection, which has not been investigated before.

6.2. Other USB Attacks and Defenses

Aside from sniffing attacks, diverse attack vectors have been explored against USB, such as *injection attacks* [7], [19], [41], [49], *data exfiltration attacks* [24], and *device masquerading attacks* [49], [79]. To defend against such threats, device authorization policies [4], [32], [44], [46], [60], [66], [68] are commonly adopted strategies. These policies vary from filtering communications to only allowing certain devices [4], [44], [66], certain interfaces within devices [32], and even certain interfaces [46], [68] to communicate with specific processes running on the host. These policies also involve authenticating devices based on their fingerprints; only devices on the "allowlist" are admitted to the system [16], [60].

Apart from the above categories, there are many other schemes to defend against USB attacks. For example, ProvUSB [65] leverages the Trusted Platform Module (TPM) to perform provenance-based attestation to thwart malware propagation and data exfiltration. FirmUSB [26] employs an attestation approach to identify any malicious functionalities in the firmware of the connected USB device. USBESAFE [37] examines the USB packet flow to detect the presence of a rogue HID device. USBFILTER [68] presents a firewall in the USB driver stack to drop/allow USB packets based on a set of rules. Johnson *et al.* [34] designed a firewall for USB packets to protect the kernel from malformed USB packets. LBM [67] leverages the eBPF packet filtering mechanism to protect against malicious peripherals within the Linux kernel. We refer readers to [69] for a more comprehensive survey on the landscape of USB threats and defenses.

6.3. Detection of Eavesdropping Devices

There have been some existing efforts on detecting eavesdropping/spying devices, including hidden cameras [10], [25], [39], [40], [57], [58], [83], voice recorders [45], [55], [77], [84], and wireless RF eavesdroppers [9], [53], [59]. Among them, the works [9], [55], [59], [84] are closest to ours for utilizing EMR to detect adversarial device's presence. Some of them assume the target device's EMR frequency composites or patterns are known *a priori* and perform the detection through device fingerprinting [9], [55], [59]. However, this approach has a limitation due to manufacturing diversity; the hardware modules and thus their emitted EMR can be different across devices. DeHiREC [84] relies on certain unique characteristics in the frequency representation of the hidden microphone's EMR for detection. In our case, sniffer's EMR spectral components are fairly complex, consisting of fundamental carrier frequencies, their harmonics, and even their combinations. It renders their frequency-domain characterization challenging. Instead of EMR's spectral characteristics, SnoopDog utilizes time-domain patterns for device detection, by examining the timing information from the bait traffic and the sniffer's responsive EMR. Hence, it nicely avoids the restrictions imposed by traditional frequency-domain fingerprinting-based device recognition methods.

7. Discussions and Conclusions

This paper presents the first attempt to detect USB bus sniffers. We developed a sniffer detection system called SnoopDog, adopting a *probe-and-detect* strategy. Bait traffic is generated to trigger sniffers to emit EMR signals correspondingly. The detection is done by examining the correlation between these two time sequences. Considering that EMR signals may be weakened by device casing or manufacturing enclosures, we designed a novel temporal folding scheme to significantly improve the EMR SNR. We built a prototype of SnoopDog and conducted a comprehensive evaluation involving 20 hubs, 12 USB devices, 10 interfering electronic devices, and 4 sniffers. SnoopDog achieves a promising detection accuracy in most tested scenarios.

In terms of usability, SnoopDog operates by having the user physically approach each target USB device with the detector. This design choice works best when the number of target devices is moderate (e.g., around 10 or less), which aligns well with most personal or office usage scenarios. In future work, we plan to explore how to improve detection efficiency in larger-scale settings with more USB devices.

References

[1] amanusk. “s-tui: Terminal-based CPU Stress and Monitoring Utility”, 2025. [Online]. Available: amanusk.github.io/s-tui/, 2025.

[2] Amazon. “Pisugar2 Plus Portable 5000 mAh UPS Lithium Battery”, 2025. [Online]. Available: amazon.com/Pisugar2-Portable-Platform-Raspberry-Accessories/dp/B08D8PPCKN, 2025.

[3] Amazon. “RTL-SDR Blog V3 R860 RTL2832U”, 2025. [Online]. Available: amazon.com/RTL-SDR-Blog-RTL2832U-Software-Defined/dp/B0BMKZCKTF, 2025.

[4] Advanced Systems International. “USB-Lock-RP”, 2025. [Online]. Available: usb-lock-rp.com/, 2025.

[5] Teledyne LeCroy. “Mercury T2”, 2025. [Online]. Available: teledynelecroy.com/protocolanalyzer/usb/mercury-t2, 2025.

[6] S. Angel, R. S. Wahby, M. Howald, J. B. Leners, M. Spilo, Z. Sun, A. J. Blumberg, and M. Walfish, “Defending against malicious peripherals with cinch,” In *25th USENIX Security Symposium*, pages 397–414, 2016.

[7] ARS-Technica. “This thumbdrive hacks computers “badusb” exploit makes devices turn “evil””, 2025. [Online]. Available: arstechnica.com/information-technology/2014/07/this-thumbdrive-hacks-computers-badusb-exploit-makes-devices-turn-evil/, 2025.

[8] Business insider. “Fake products sold by places like walmart or amazon hold risks of everything from cyanide to rat droppings”, 2025. [Online]. Available: businessinsider.com/how-to-find-fake-products-online-shopping-amazon-ebay-walmart-2018-3, 2025.

[9] A. Chaman, J. Wang, J. Sun, H. Hassanieh, and R. R. Choudhury, “ Ghostbuster: Detecting the presence of hidden eavesdroppers,” In *Proceedings of the 24th annual international conference on mobile computing and networking (MobiCom)*, pages 337–351, 2018.

[10] Y. Cheng, X. Ji, T. Lu, and W. Xu, “Dewicam: Detecting hidden wireless cameras via smartphones,” In *Proceedings of the 2018 Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 1–13, 2018.

[11] J. Choi, H. Y. Yang, and D. H. Cho, “Tempest comeback: A realistic audio eavesdropping threat on mixed-signal socs,” In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1085–1101, 2020.

[12] Chron. “Flash drive marketing ideas”, 2025. [Online]. Available: smallbusiness.chron.com/fiveyear-anniversary-business-promotion-ideas-13884.html, 2025.

[13] Compaq, Digital Equipment Corporation, IBM PC Company, Intel, Microsoft, NEC, and Northern Telecom, *Universal Serial Bus Specification, Rev 1.0.*, January 1996.

[14] Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, and Philips, *Universal Serial Bus Specification, Rev 2.0.*, April 2000.

[15] Compaq, Intel, Microsoft, and NEC, *Universal Serial Bus Specification, Rev 1.1.*, September 1998.

[16] P. Cronin, X. Gao, H.Wang, and C. Cotton. Timeprint: Authenticating usb flash drives with novel timing fingerprints. In *IEEE Symposium on Security and Privacy (SP)*, pages 1002–1017, 2022.

[17] D. Spill, “Usb proxy”, 2025. [Online]. Available: github.com/usb-tools/, 2025.

[18] R. Dumitru, M. Beaumont, B. Hopkins, and S. Windows, “ Usb proxy,” In *Proceedings of the 2023 Australasian Computer Science Week*, pages 122–125. 2023.

[19] R. Dumitru, D. Genkin, A. Wabnitz, and Y. Yarom, “The impostor among US (B):Off-Path injection attacks on USB communications,” In *32nd USENIX Security Symposium*, pages 5863– 5880, 2023.

[20] ECS Inc, “Ecs-240-20-30b-du”, 2025. [Online]. Available: ecsxtal.com/products/crystals/surface-mount-crystals/ecs-240-20-30b-du/, 2025.

[21] Great Scott Gadgets, “Cynthion”, 2025. [Online]. Available: greatscottgadgets.com/cynthion/, 2025.

[22] Great Scott Gadgets, “Luna-great scott gadgets.”, 2025. [Online]. Available: greatscottgadgets.com/tags/luna/, 2025.

[23] Great Scott Gadgets, “Sniffing ps5 controller packets with cynthion.”, 2025. [Online]. Available: greatscottgadgets.com/2024/11-27-sniffing-ps5-controller-packets-with-cynthion/, 2025.

[24] M. Guri, M. Monitz, and Y. Elovici, “Usbee: Air-gap covert-channel via electromagnetic emission from usb,” In *14th Annual Conference on Privacy, Security and Trust (PST)*, pages 264–268, 2016.

[25] Y.He, Q. He, S. Fang, and Y. Liu, “Motioncompass: pinpointing wireless camera via motion-activated traffic,” In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 215–227, 2021.

[26] G. Hernandez, F. Fowze, D. Tian, T. Yavuz, and K. R. B. Butler, “Firmusb: Vetting usb device firmware using domain informed symbolic execution,” In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2245–2262, 2017.

[27] Apple Inc., HP Inc., Intel Corporation, Microsoft Corporation, Renesas Corporation, STMicroelectronics, and Texas Instruments, *Universal Serial Bus 4 (USB4™) Specification, Version 1.0.*, August 2019.

[28] Hewlett Packard Company, Intel Corporation, Microsoft Corporation, NEC Corporation, ST-NXP Wireless, and Texas Instruments. *Universal Serial Bus 3.0 Specification, Rev 1.0.*, November 2008.

[29] Infineon, “Uez-usb fx2lp usb microcontroller high-speed usb peripheral controller,” 2025. [Online]. Available: infineon.com/dgdl/Infineon-CY7C68013A_CY7C68014A_CY7C68015A_CY7C68016A, 2025.

[30] Intel, “Cyclone® iv featured documentation - quick links guide,” 2025. [Online]. Available: intel.com/content/www/us/en/docs/programmable/767845/current/cyclone-iv-featured-documentation-quick.html, 2025.

[31] Intel Corporation. *eXtensible Host Controller Interface for Universal Serial Bus (xHCI) Requirements Specification*, Revision 1.2, May 2019.

[32] Ivanti, “Ivanti device control,” 2025. [Online]. Available: ivanti.com/en-au/products/device-control, 2025.

[33] J. R. Jacobs, “Measuring the effectiveness of the usb flash drive as a vector for social engineering attacks on commercial and residential computer systems,” *Embry Riddle Aeronautical University*, 2011.

[34] P. C. Johnson, S. Bratus, and S. W. Smith, “Protecting against malicious bits on the wire: Automatically generating a usb protocol parser for a production kernel,” In *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC)*, pages 528–541, 2017.

[35] Keelog, “Keygrabber usb,” 2025. [Online]. Available: keelog.com/, 2025.

[36] KeyGhost, “Keyghost usb keylogger,” 2025. [Online]. Available: keyghost.com/usb-keylogger.htm, 2025.

[37] A. Kharraz, B. L. Daley, G. Z. Baker, W. Robertson, and E. Kirda, “Usbesafe: An end-point solution to protect against usb-based attacks,” In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, pages 89–103, 2019.

[38] D.Kierznowski, “Badusb 2.0: Usb man in the middle attacks,” *Retrieved from RoyalHolloway*, 2016.

[39] Z. Li, Z. Xiao, Y. Zhu, I. Pattarachanyakul, B. Y. Zhao, and H. Zheng, “Adversarial localization against wireless cameras,” In *Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications*, pages 87–92, 2018.

[40] Z. Liu, F. Lin, C. Wang, Y. Shen, Z. Ba, L. Lu, W. Xu, and K. Ren, “Camradar: Hidden camera detection leveraging amplitude-modulated sensor images embedded in electromagnetic emanations,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2023.

[41] H. Lu, Y. Wu, S. Li, Y. Lin, C. Zhang, and F. Zhang, "Badusb-c: Revisiting badusb with type-c," In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 327–338, 2021.

[42] PL Lui, "Passive intermodulation interference in communication systems," *Electronics & Communication Engineering Journal*, 2(3):109–118, 1990.

[43] Microchip, "Usb3300 data sheet," 2025. [Online]. Available: ww1.microchip.com/downloads/en/DeviceDoc/00001783C.pdf, 2025.

[44] Microsoft, "Microsoft windows embedded 8.1 industry usb filter (industry 8.1)," 2025. [Online]. Available: msdn.microsoft.com/en-us/library/dn449350(v=winembedded.82).aspx, 2025.

[45] S. Mirzamohammadi and A. A. Sani, "Viola: Trustworthy sensor notifications for enhanced privacy on mobile systems," In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 263–276, 2016.

[46] H. Mohammadmoradi and O. Gnawali, "Making whitelisting-based defense work against badusb," In *Proceedings of the 2nd International Conference on Smart Digital Environment*, pages 127–134, 2018.

[47] M. Neugschwandtner, A. Beitler, and A. Kurmus, "A transparent defense against usb eavesdropping attacks," In *Proceedings of the 9th European Workshop on System Security*, pages 1–6, 2016.

[48] S. Neuner, A. G Voyiatzis, S. Fotopoulos, C. Mulliner, and E. R. Weippl, "Usblock: Blocking usb-based keypress injection attacks," In *Data and Applications Security and Privacy XXXII: 32nd Annual IFIP WG 11.3 Conference (DBSec 2018)*, pages 278–295, 2018.

[49] K. Nohl and J. Lell, "Badusb-on accessories that turn evil," *Black Hat USA*, 1(9):1–22, 2014.

[50] NY Times-Wirecutter, "Welcome to the era of fake products," 2025. [Online]. Available: nytimes.com/wirecutter/blog/amazon-counterfeit-fake-products/, 2025.

[51] J. Oberg, W. Hu, A. Irturk, M. Tiwari, T. Sherwood, and R. Kastner, "Information flow isolation in i2c and usb," In *Proceedings of the 48th Design Automation Conference*, pages 254–259, 2011.

[52] Openviszla, "Ft2232h-based usb sniffer," 2025. [Online]. Available: github.com/openvizsla/ov_ftdi, 2025.

[53] S. Park, L. E. Larson, and L. B. Milstein, "An rf receiver detection technique for cognitive radio coexistence," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(8):652–656, 2010.

[54] P. Sewers, "Us govt. plant usb sticks in security study, 60the bait," 2025. [Online]. Available: thenextweb.com/insider/2011/06/28/us-govt-plant-usb-sticks-in-security-study-60-of-subjects-take-the-bait/, 2025.

[55] S. Ramesh, G. S. Hadi, S. Yang, M. C. Chan, and J. Han, "Ticktock: Detecting microphone status in laptops leveraging electromagnetic leakage of clock signals," In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2475–2489, 2022.

[56] Raspberry Pi, "Raspberry pi 4b," 2025. [Online]. Available: raspberrypi.com/products/raspberry-pi-4-model-b/, 2025.

[57] M. Salman, N. Dao, U. Lee, and Y. Noh, "Csi: Despy: Enabling effortless spy camera detection via passive sensing of user activities and bitrate variations," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(2):1–27, 2022.

[58] S. Sami, S. R. X. Tan, B. Sun, and J. Han, "Lapd:Hidden spy camera detection using smartphone time-of-flight sensors," In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 288–301, 2021.

[59] C. Shen and J. Huang, "Earfisher: Detecting wireless eavesdroppers by stimulating and sensing memory emr," In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 873–886, 2021.

[60] Z. Y. S. Liao, H. Chen, "Securityhub: Electromagnetic fingerprinting usb peripherals using backscatter-assisted commodity hardware," In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2024.

[61] SMSC-Microchip, "An 19.17 - ulpi design guide - smsc.," 2025. [Online]. Available: ww1.microchip.com/downloads/en/AppNotes/en562704.pdf, 2025.

[62] StarTech.com., "Usb to vga adapter - 1920x1200 - taa," 2025. [Online]. Available: startech.com/en-us/display-video-adapters/usb2vgae3, 2025.

[63] Y. Su, D. Genkin, D. Ranasinghe, and Y. Yarom, "USB snooping made easy: crosstalk leakage attacks on USB hubs," In *26th USENIX Security Symposium*, pages 1145–1161, 2017.

[64] TEquipment, "Tekbox tbps01 - emc near field probe set," 2025. [Online]. Available: tequipment.net/TekBox/TBPS01/EMI-Accessories/, 2025.

[65] D.Tian, A. Bates, K. R. B. Butler, and R. Rangaswami, "Provusb: Block-level provenance-based data protection for usb storage devices," In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 242–253, 2016.

[66] D. J. Tian, A. Bates, and K. Butler, "Defending against malicious usb firmware with goodusb," In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC)*, pages 261–270, 2015.

[67] D. J. Tian, G. Hernandez, J. I Choi, V. Frost, P. C Johnson, and K. R. B. Butler, "Lbm: A security framework for peripherals within the linux kernel," In *IEEE Symposium on Security and Privacy (SP)*, pages 967–984, 2019.

[68] D. J. Tian, N. Scaife, A. Bates, K. Butler, and P. Traynor, "Making USB great again with USBFILTER," In *25th USENIX Security Symposium*, pages 415–430, 2016.

[69] J. Tian, N. Scaife, D. Kumar, M. Bailey, A. Bates, and K. Butler, "Sok: "plug & pray" today–understanding usb insecurity in versions 1 through c," In *IEEE Symposium on Security and Privacy (SP)*, pages 1032–1047, 2018.

[70] M. Tischer, Z. Durumeric, S. Foster, S. Duan, A. Mori, E. Bursztein, and M. Bailey, "Users really do plug in usb drives they find," In *IEEE Symposium on Security and Privacy*, pages 306–319, 2016.

[71] TotalPhase, "Beagle usb 480 power protocol analyzer - ultimate edition," 2025. [Online]. Available: totalphase.com/products/beagle-usb480-power-ultimate/, 2025.

[72] USA Today, "$1.7 trillion in fake goods: That's alibaba's uphill battle," 2025. [Online]. Available: usatoday.com/story/tech/news/2017/01/16/alibaba-cracks-down-17-trillion-fake-goods-market/96633850/, 2025.

[73] USB-Implementers Forum, "Designing a robust usb serial interface engine (sie)," 2025. [Online]. Available: usb.org/document-library/designing-robust-usb-serial-interface-engine-sie, 2025.

[74] USB-Implementers Forum, "Usb 3.1 debug class specification for debug devices, rev. 1.0. ," 2025. [Online]. Available: usb.org/document-library/usb-31-legacy-cable-and-connector-revision-10, 2025.

[75] USB-Implementers Forum, "Usb-if statement regarding usb security," 2025. [Online]. Available: usb.org/press/USBIF, 2025.

[76] USB Memory Direct, "How to use usb drives for marketing," 2025. [Online]. Available: usbmemorydirect.com/blog/usb-marketing/, 2025.

[77] Veronica Valeros and Sebastian Garcia, "Spy vs. spy: A modern study of microphone bugs operation and detection," *Chaos Computer Club eV*, 2017.

[78] Which? research, "Where did all the usb ports go?," 2025. [Online]. Available: which.co.uk/news/article/where-did-all-the-usb-ports-go-aMXwM1I0WvA3, 2025.

[79] Wikipedia, "Ant catalog," 2025. [Online]. Available: en.wikipedia.org/wiki/ANT_catalog, 2025.

[80] Yahoo finance, "Usb hub market to gain usd 8.4 bn by 2032 — increasing demand for multi-device connectivity and faster data transfer speeds.," 2025. [Online]. Available: finance.yahoo.com/news/usb-hub-market-gain-usd-093000228.html, 2025.

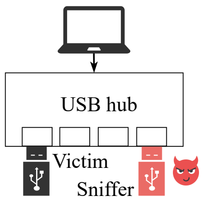
[81] Yahoo finance, "Usb devices market forecast to 2027," 2025. [Online]. Available: uk.finance.yahoo.com/news/usb-devices-market-forecast-2027-110100432.html, 2025.

[82] YouTube, "Why amazon has so many counterfeit goods," 2025. [Online]. Available: youtube.com/watch?v=wfPM3i9NIHM, 2025.

[83] Z. Yu, Z. Li, Y. Chang, S. Fong, J. Liu, and N. Zhang, "Heatdecam: Detecting hidden spy cameras via thermal emissions," In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 3107–3120, 2022.

[84] R. Zhou, X. Ji, C. Yan, Y.C. Chen, W. Xu, and C. Li, "Dehirec: Detecting hidden voice recorders via adc electromagnetic radiation," In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 3113–3128, 2023.

Appendix A. Scenarios Susceptible to Sniffing Attacks



Config. #	1	2	3	4
Hub	2.0	2.0	3.x	3.x
Device	2.0	3.x	2.0	3.x
Safe?	✗	✗	✗	✓

Figure 11: Left: The setting of the case study 1. Right: Whether the victim device is free from the sniffing attack under four possible configurations.

For the USB 1.x/2.0, downstream traffic is broadcast across the hub, rendering it accessible to all devices connected to the hub, thus exposing all downstream communications to potential eavesdropping by sniffers, as outlined in §2.2. Despite downstream traffic being unicast in USB 3.x and beyond, the risk of sniffing attacks has not been completely eliminated. As highlighted by several studies [19], [47], [63], vulnerabilities still exist, particularly when USB 1.x/2.0 devices coexist with that of later versions, due to USB's backward compatibility feature. Nonetheless, no prior works specifically identified the conditions in which the coexistence of different USB versions is susceptible to sniffing attacks. We thus carry out case studies trying to bridge this gap. More importantly, the vulnerable scenarios identified herein provide essential guidance for setting up scenarios to evaluate SnoopDog's effectiveness.

To investigate the vulnerability of coexistence of different USB versions (USB 2.0 and 3.x particularly), we conduct a series of case studies, reflecting typical USB device and hub connection scenarios encountered in everyday use. The following are two representative ones. We intend to draw a conclusion in the end.

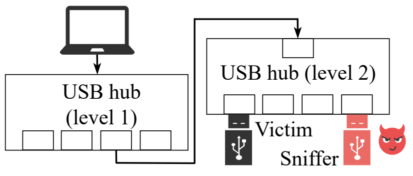
Case study 1: One hub in sequence. We start from a basic setting, where the host PC is extended with one USB hub, which is connected with a victim device and a sniffer, as shown in Figure 11. Following prior work [47], we use the Beagle USB 480 protocol analyzer [71] as the sniffer and a USB flash drive as the victim. We execute a program script at the host to transmit a file with the content "Annual Computer Security Applications Conference 2025" addressed to the victim. The analyzer is set to `monitor mode` to observe all bus traffic, just like a sniffer. Figure 13a shows the screenshot of the analyzer's GUI when the attack is successful. Apparently, the analyzer is able to capture and access the plaintext of the transferred file. The attack succeeds if any element, including the hubs and the victim itself, in the downstream traffic chain adopts USB 2.0 or an earlier version. It includes configurations # 1-3 in case study 1. As a comparison, we also show in Figure 13b the GUI screenshot when the victim device is free from the sniffing attack. The analyzer cannot access any data on the USB bus. It includes configuration # 4 in case study 1, where the downstream traffic is unicast.

Case study 2: Multiple hubs in sequence. We further extend Case study 1 to more than one hub concatenated in sequence, as shown in Figure 12. This is a common approach for users to increase the number of PC's USB ports. For simplicity, we only consider two hubs in sequence here, wherein the victim device and the sniffer are connected to the same hub. We list the attack results under all tested configurations in Figure 12. Similarly, all configurations are vulnerable, except when all three devices (i.e., two hubs and the victim device) follow USB 3.x.

Discussions. The two case studies described above serve as representative examples for the conditions under which coexistence of different USB versions becomes susceptible to sniffing attacks. Numerous other configurations could also be vulnerable. For instance, scenarios involve more than two USB hubs linked in sequence or cases where the victim and the sniffer are connected to separate hubs. Due to the limited space, we did not enumerate them all here. Nevertheless, we draw the following conclusions from our observations: ***The victim device is vulnerable to sniffing attacks as long as any entity, including the hubs and the victim itself, in the downstream traffic chain adopts USB 1.x or 2.0.*** Here a "chain" starts from the host PC and ends at the victim device. These vulnerabilities stem from the backward compatibility features inherent in USB protocols. Hubs of USB 3.x and later versions are designed with USB 1.x/2.0 data lanes running parallel to maintain this compatibility. Consequently, when a USB 1.x/2.0 device is connected to a USB hub in 3.x or later versions, it triggers the activation of these USB 1.x/2.0 lanes for communication. As a result, despite the presence of advanced USB technology, the downstream traffic reverts to the broadcast mode of USB 1.x/2.0.

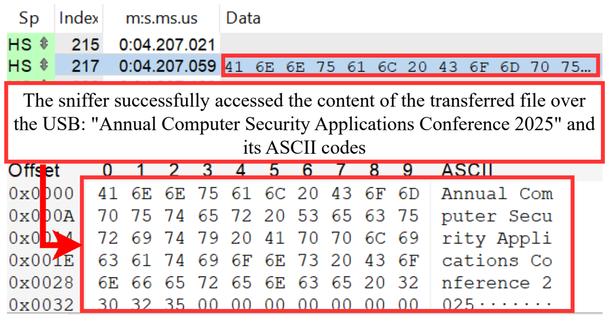
Appendix B. Spectrum Analysis of Sniffer's EMR

To precisely identify the frequency components of sniffer's EMR, we employ a Tektronix MDO34 oscilloscope to perform spectrum analysis. Figure 14 exhibits the measure-



Configuration #	1	2	3	4	5	6	7	8
USB hub (level 1)	2.0	2.0	2.0	2.0	3.x	3.x	3.x	3.x
USB hub (level 2)	2.0	2.0	3.x	3.x	2.0	2.0	3.x	3.x
USB device	2.0	3.x	2.0	3.x	2.0	3.x	2.0	3.x
Safe?	✗	✗	✗	✗	✗	✗	✗	✓

Figure 12: Left: The setting of the case study 2. Right: Whether the victim device is free from the sniffing attack under eight possible configurations.

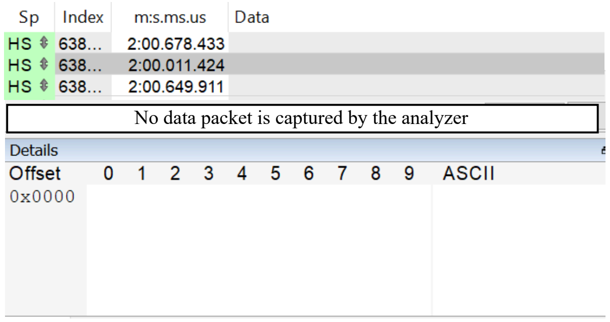


```
Sp   Index  m:s.ms.us     Data
HS   215    0:04.207.021
HS   217    0:04.207.059  41 6E 6E 75 61 6C 20 43 6F 6D 70 75...
```

The sniffer successfully accessed the content of the transferred file over the USB: "Annual Computer Security Applications Conference 2025" and its ASCII codes

```
Offset  0  1  2  3  4  5  6  7  8  9   ASCII
0x0000  41 6E 6E 75 61 6C 20 43 6F 6D  Annual Com
0x000A  70 75 74 65 72 20 53 65 63 75  puter Secu
0x0014  72 69 74 79 20 41 70 70 6C 69  rity Appli
0x001E  63 61 74 69 6F 6E 73 20 43 6F  cations Co
0x0028  6E 66 65 72 65 6E 63 65 20 32  nference 2
0x0032  30 32 35 00 00 00 00 00 00 00  025·······
```

(a)



```
Sp   Index  m:s.ms.us    Data
HS   638...  2:00.678.433
HS   638...  2:00.011.424
HS   638...  2:00.649.911
```

No data packet is captured by the analyzer

```
Details
Offset  0  1  2  3  4  5  6  7  8  9   ASCII
0x0000
```

(b)

Figure 13: Screenshot of the analyzer's GUI. (a) under a successful attack, (b) no attack

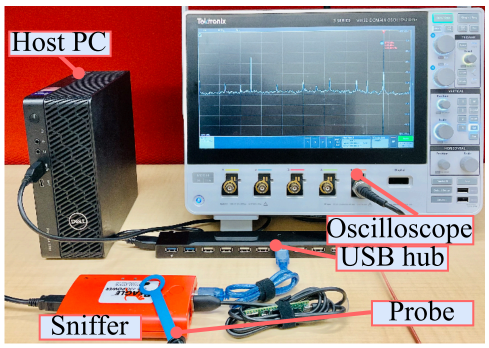


Figure 14: Measurement setup for spectrum analysis of sniffer's EMR.

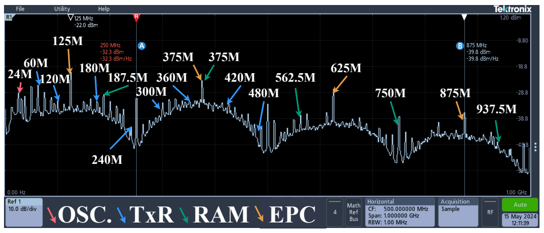
ment setup, where a sniffer is connected to a USB hub that is further connected to a host PC.

Figure 15a annotates some of the prominent frequency components from EMR, which are classified into two categories: a) the fundamental carrier frequencies from major circuit modules, such as RAM, TxR, EPC, and onboard oscillator, and b) their harmonics. Specifically, carrier frequencies of RAM consist of 187.5 MHz, 375 MHz, 562.5 MHz, 750 MHz, $\cdots$; the clock frequencies of the external on-board oscillator and its harmonics consist of 24 MHz, 48 MHz, 72 MHz, $\cdots$; the TxR internal reference clock and its harmonics consist of 60 MHz, 120 MHz, 180 MHz, $\cdots$; those of EPC consist of 125 MHz, 250 MHz, 375 MHz, $\cdots$. They all follow the expression of $n \times f_c$ where f_c is the fundamental carrier frequency. The sources are confirmed by comparing experimental measures with the numbers provided in the related datasheets [20], [30], [43], [61]. We did not exhaustively annotate all the prominent frequency components, including the ones generated by the effect of *carrier inter-modulation & coupling* though.

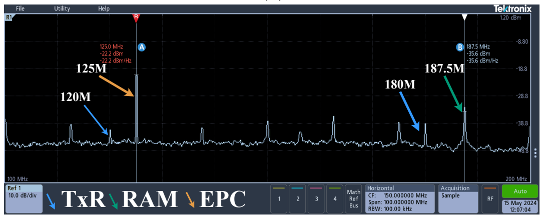
Figure 15b and 15c show the sniffer's power spectrum between 100-200 MHz under two modes, i.e., active and standby. The former means the sniffer is copying data from the bus, whereas the latter means there is no ongoing transmission and thus the sniffer is inactive. Among the prominent frequency components, the ones associated with TxR, EPC, and oscillator are present in both modes, while that associated with RAM only exists when the sniffer is active. It indicates that RAM only works when the sniffer is active, i.e., capturing and writing data from the USB bus to the RAM. The corresponding operation produces EMR. This phenomenon complies with the discussion in §2.1.

Discussions. With the results from Figure 15b and 15c, we were asking whether it is possible to leverage the EMR signal from the RAM of a sniffer for the detection: If a target device's EMR aligns with the bait traffic in timing, it is a sniffer; otherwise, it is not. While this concept is theoretically sound, it presents a challenge. The detector needs to know the RAM's operational frequency of the target device in advance so that it can tune the probe to that specific frequency band for signal capturing. However, this is a strong assumption in real-world scenarios due to hardware manufacturing diversity.

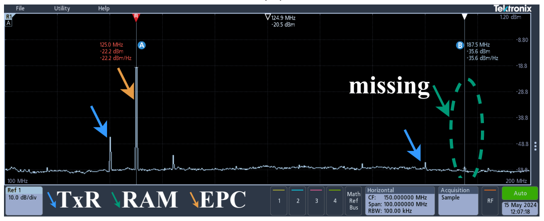
Fortunately, as discussed in §3.1, the sniffer's EMR signal aligns well with the bait traffic in timing over a broad spectrum band (e.g., 100-200 MHz). It means that instead of tuning to RAM's operation frequency, we can scan any sub-band of meaningful bandwidth from the range of MHz to GHz to collect evidence. This nice property is attributed to the non-ideal and non-linear characteristics of



(a)

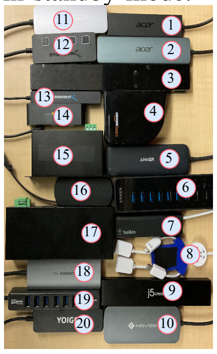


(b)



(c)

Figure 15: Frequency components of sniffer's EMR signals. (a) Zoom-out: [0, 1 GHz]. (b) Zoom-in: [100 MHz, 200 MHz]. (c) Zoom-in: [100 MHz, 200 MHz] when the sniffer is in standby mode.


(a) Hubs.

(b) USB devices.

(c) Sniffers.

Figure 16: Devices used in the evaluation.

sniffer's circuits and the effect of carrier inter-modulation & coupling, which cause the fundamental carrier frequency to spread across a wide band from MHz to GHz.

Appendix C. Statistical Analysis of USB Traffic

Table 8 gives the statistical analysis of the transmission metrics, with their relevant parameters. The result is derived from four common USB applications: *file transfer*, *audio streaming*, *video streaming*, and *online activity*. A total of 11 USB peripherals are examined. They cover the products in both the USB 2.0 and 3.x standards from representative manufacturers. The result is presented in the format of [minimum, medium, maximum] indicating the minimum, medium, and maximum values of that particular metric observed in the analytic study.

TABLE 8: Statistical analysis of common USB applications. ●: Device(s) in USB 2.0; ○: Device(s) in USB 3.x; 📶: Online activity; 🎧: Audio streaming; 🖥: Video streaming; 📄: File transfer.

ID/ #	Device	USB version	Service	Metrics				
				Packets per microframe	Packet-size (B)	Queue time (μs)	Service latency (ms)	Throughput (Mbps)
1	BrosTrend AC1200	○		[0,**11**,14]	[0,**512**,512]	[1,10,125]	[0.12,**2.75**,126]	[36.8,**64.5**,89.9]
2	TP-Link ARCHER T2UB Nano	●	📶	[0,**9**,13]	[0,**512**,512]	[1,**11**,125]	[0.12, **2.91**,125.9]	[15.71,**23.18**,40.23]
3	TP-Link TL-WN722N	●		[0, **3**,14]	[0,**512**,512]	[2,**10**,125]	[0.12,**1.57**,184]]	[9.7,**12.6**,16.9]
4	Logitech Headset H340	●		[0,**1**,1]	[0,**122**,180]	[125,**125**,125]	[0.25,**1**,8.001]	[1.73,**1.74**,1.77]
5	Logitech Headset H390	●	🎧	[0,**1**,1]	[0,**124**,193]	[125,**125**,125]	[0.25,**1**,1.13]	[1.79,**1.79**,1.83]
6	StarTech.com 7.1 Sound Card	●		[0,**1**,1]	[0,**128**,192]	[125,**125**,125]	[0.12,**1**,1.02]	[1.79, **1.80**, 1.82]
7	Plugable VGA Adapter	●	🖥	[0,**11**,13]	[0,**512**,512]	[2,**20**,125]	[0.12,**1.88**,51.4]	[67.41,**103.87**,135.8]
8	StarTech.com VGA Adapter	●		[0,**11**,12]	[0,**512**,512]	[9,**10**,125]	[0.12,**0.25**,58.6]	[81,**93.9**,109.3]
9	SanDisk Cruzer Blade	●		[0,**10**,12]	[0,**512**,513]	[9,**10**,125]	[0.12,**0.28**,786.6]	[41.5,**69.1**,164.1]
10	SanDisk Ultra Dual Drive Luxe	○	📄	[0,**10**,11]	[31,**512**,512]	[9,**11**,125]	[0.12, **0.37**,121.1]	[12.6,**142.8**,170.1]
11	Verbatim ToughMax	●		[0,**11**,11]	[31,**512**,512]	[10,**11**,125]	[0.12,**0.17**,849.3]	[0.52,**5.78**,124.5]

(1)*Packets per microframe:* The number of packets present in one USB microframe.; (2)*Packet-size:* The size of each USB packet.; (3)*Queue time:* This represents the amount of time that the packets spend in a device's hardware buffer before being processed. Lower values denote quicker response times.; (4)*Service latency:* This represents the incurred latency for different service types. It is a combination of network latency, user interaction, host controller management, etc.; (5)*Throughput:* The amount of data transfer in each second.