

POWER-SUPPLaY: Leaking Data from Air-Gapped Systems by Turning the Power-Supplies Into Speakers

Mordechai Guri

Ben-Gurion University of the Negev, Israel

Cyber-Security Research Center

gurim@post.bgu.ac.il

Air-Gap research page: <http://www.covertchannels.com>

Demo video: <http://www.covertchannels.com>

Abstract—It is known that attackers can exfiltrate data from air-gapped computers through their speakers via sonic and ultrasonic waves. To eliminate the threat of such acoustic covert channels in sensitive systems, audio hardware can be disabled and the use of loudspeakers can be strictly forbidden. Such audio-less systems are considered to be *audio-gapped*, and hence immune to acoustic covert channels.

In this paper, we introduce a technique that enable attackers leak data acoustically from air-gapped and audio-gapped systems. Our developed malware can exploit the computer power supply unit (PSU) to play sounds and use it as an out-of-band, secondary speaker with limited capabilities. The malicious code manipulates the internal switching frequency of the power supply and hence controls the sound waveforms generated from its capacitors and transformers. Our technique enables producing audio tones in a frequency band of 0-24kHz and playing audio streams (e.g., WAV) from a computer power supply without the need for audio hardware or speakers. Binary data (files, keylogging, encryption keys, etc.) can be modulated over the acoustic signals and sent to a nearby receiver (e.g., smartphone). We show that our technique works with various types of systems: PC workstations and servers, as well as embedded systems and IoT devices that have no audio hardware at all. We provide technical background and discuss implementation details such as signal generation and data modulation. We show that the POWER-SUPPLaY code can operate from an ordinary user-mode process and doesn't need any hardware access or special privileges. Our evaluation shows that using POWER-SUPPLaY, sensitive data can be exfiltrated from air-gapped and audio-gapped systems from a distance of five meters away at a maximal bit rates of 50 bit/sec.

I. INTRODUCTION

Air-gapped computers are kept isolated from the Internet or other less secure networks. Such isolation is often enforced when sensitive or confidential data is involved, in order to reduce the risk of data leakage. Military networks such as the Joint Worldwide Intelligence Communications System (JWICS) [1], as well as networks within financial organizations, critical infrastructure, and commercial industries [2], [3], are known to be air-gapped due to the sensitive data they handle.

Despite the high degree of isolation, even air-gapped networks can be breached using complex attack vectors such as supply chain attacks, malicious insiders, and deceived

insiders. Famous air-gap breaching cases include Stuxnet [4] and Agent.BTZ [5], but other incidents have also been reported [6], [7]. In 2018, The US Department of Homeland Security accused Russian government hackers of penetrating America's power utilities [8]. Due to reports in the Washington Post in November 2019, the Nuclear Power Corporation of India Limited (NPCIL) confirmed that the Kudankulam Nuclear Power Plant suffered a cyber-attack earlier that year [9].

A. Air-Gap Covert Channels

While the *infiltration* of such networks has been shown to be feasible, the *exfiltration* of data from non-networked computers or those without physical access is considered a challenging task. Over the years, different types of out-of-band covert channels have been proposed, exploring the feasibility of data exfiltration through an air-gap. *Electromagnetic* methods that exploit electromagnetic radiation from different components of the computer are likely the oldest kind of air-gap covert channel researched [10], [11], [12], [13], [14]. Other types of *optical* [15], [16] and *thermal* [17] out-of-band channels have also been studied. *Acoustic* exfiltration of data using inaudible sound has also been explored in many studies [18], [19], [20], [21]. The existing acoustic methods suggest transmitting data through the air-gap via high frequency sound waves generated by computer loudspeakers. Note that existing acoustic covert channels rely on the presence of audio hardware and loudspeakers in the compromised computer.

B. Audio-Gap: Speaker-less, Audio-less Systems

To cope with acoustic covert channels, common practices and security policies strictly prohibit the use of speakers on sensitive computers, in order to create a so-called 'audio-gapped' environment [22], [23]. As an additional defensive measure, the audio chip may be disabled in the UEFI/BIOS to cope with the accidental attachment of loudspeakers to the *line-out* connectors. Obviously, disabling the audio hardware and keeping speakers disconnected from computers can effectively mitigate the acoustic covert channels presented thus far [24].

C. Our Contribution

In this paper, we introduce a new acoustic channel which doesn't require speakers or other audio related hardware. We show that malware running on a PC can exploit its power supply unit (PSU) and use it as an out-of-band speaker with limited capabilities. The malicious code intentionally manipulates the internal *switching frequency* of the power supply and hence controls the waveform generated from its capacitors and transformers. This technique enables playing audio streams from a computer even when audio hardware is disabled and speakers are not present. We show that our technique works with various types of systems: PC workstations and servers, as well as embedded systems and IoT devices that have no audio hardware. Binary data can be modulated and transmitted out via the acoustic signals. The acoustic signals can then be intercepted by a nearby receiver (e.g., a smartphone), which demodulates and decodes the data and sends it to the attacker via the Internet.

The proposed method has the following unique characteristics:

- **Requires no audio hardware.** The method allows malware to play audible and inaudible sounds from systems which are completely audio-gapped (e.g., speakers are disconnected) or systems that doesn't have any type of audio hardware (e.g., embedded devices).
- **Requires no special privileges.** The method doesn't require special privileges or access to hardware resources. The transmitting code can be initiated from an ordinary user-space process and is highly evasive.

The rest of this paper is organized as follows: The attack model is discussed in Section II. Related work is presented in Section III. Background on the power supply acoustics is provided in Section IV. Section V and Section VII, respectively, contain details on The transmitter and receiver. Section VIII describes the analysis and evaluation. Countermeasures are discussed in Section IX. We conclude in Section X.

II. ATTACK MODEL

The capability of generating acoustic tones through power supplies can be considered a general contribution to the field of acoustic covert channels, regardless of its connection to the air-gap. However, in this paper, we investigate it as a method of exfiltrating information from air-gapped, audio-gapped systems. Similar to other covert communication channels, the adversarial attack model consists of a transmitter and a receiver. Typically in such scenarios, the transmitter is a computer, and the receiver is a nearby mobile phone belonging to an employee or visitor (Figure 1).

1) *Infection phase*: In a preliminary stage, the transmitter and receiver are compromised by the attacker. Infecting highly secure networks can be accomplished, as demonstrated by the attacks involving Stuxnet [4] and Agent.Btz [5], and other attacks [25], [6], [7]. In our case, the infected computer must be equipped with an internal power supply which exists in virtually every computerized system today. In addition, the

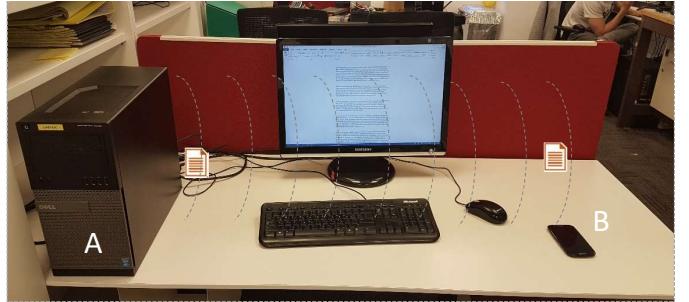


Fig. 1: Exfiltration scenario: malware within the infected air-gapped, audio-gapped (speaker-less) computer (A) leaks a file through inaudible sound waves played through the power supply. The file is received by a nearby smartphone (B).

mobile phones of employees are identified, possibly by social engineering techniques. The employees are assumed to carry their mobile phones around the workplace. These devices are then infected, either online, by exploiting a device's vulnerabilities, or by physical contact when possible. Infecting a mobile phone can be accomplished via different attack vectors, using emails, SMS/MMS, malicious apps, malicious websites, and so on [26], [27], [28], [29], [30].

2) *Exfiltration phase*: In the exfiltration phase, the malware in the compromised computer gathers sensitive data of interest. The data can be files, keystroke logging, credentials (e.g., passwords), or encryption keys. The malware then modulates and transmits the data using the acoustic sound waves emitted from the computer's power supply (Figure 2). A nearby infected mobile phone detects the transmission, demodulates and decodes the data, and transfers it to the attacker via the Internet using mobile data or Wi-Fi. Note that in this paper, we demonstrate the attack model using a mobile phone receiver, a device which is commonly located in the vicinity of a computer. Other types of receivers are devices with internal or external microphones such as laptops and desktop workstations.

III. RELATED WORK

Covert channels in *networked* environments have been widely discussed in professional literature for years [31], [32], [33]. In these covert channels, attackers may hide data within existing network protocols (such as HTTPS, SMTP and DNS), conceal data within images (stenography), encode it in packet timings and so on.

Our work focuses on the challenge of leaking data from computers that have no network connectivity (air-gapped computers). Over the years different types of out-of-band covert channels have been proposed, allowing the attacker to bridge the air-gap isolation. The methods can be mainly categorized into electromagnetic and magnetic, optical, thermal, and acoustic covert channels.

Electromagnetic emissions are probably the oldest type of methods that have been explored academically with regard to air-gap communication. In a pioneer work in this field, Kuhn

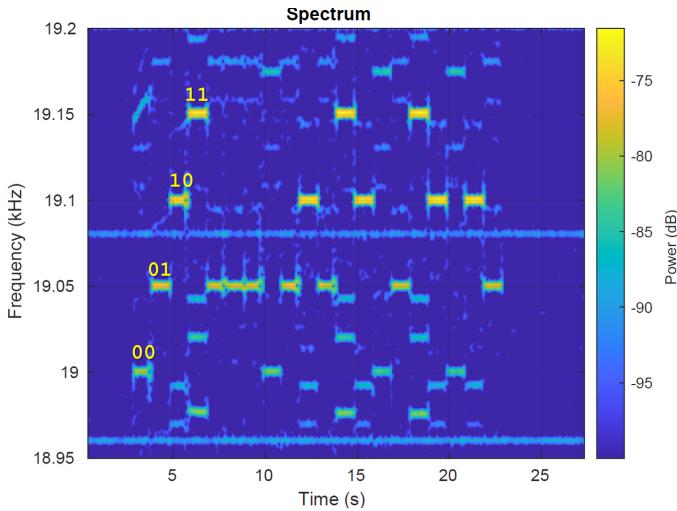


Fig. 2: The information has been exfiltrated via covert ultrasonic sound signals played from the power supply. As can be seen in the spectrogram, four different frequencies are used for modulation.

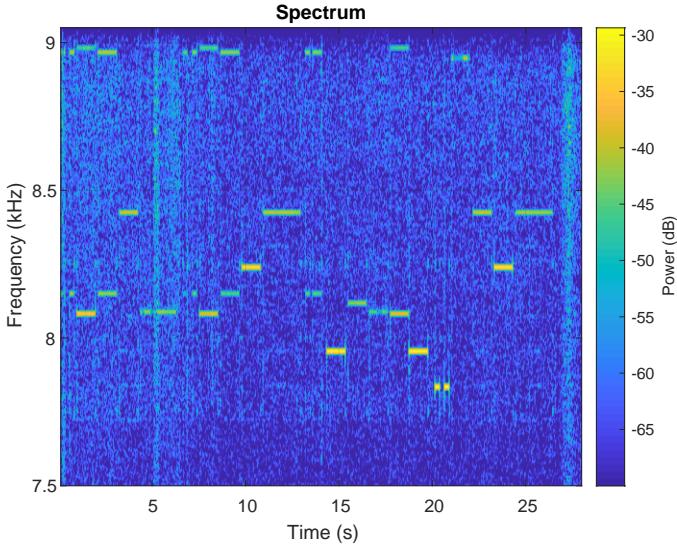


Fig. 3: Part of the song 'Happy Birthday' as played through the power supply.

and Anderson [11] discuss hidden data transmission using electromagnetic emissions from video cards. Back in 2001, Thiele [34] utilized the computer monitor to transmit radio signals to a nearby AM radio receiver. AirHopper malware [10], [2] introduced in 2014 by Guri et al, exploits video card emissions to bridge the air-gap between isolated computers and nearby mobile phones via FM radio signals. In a similar manner, GSMem [14], Funthenna [35] and USBee [36], introduce attack scenarios in which attackers use different sources of electromagnetic radiation on a computer's motherboard, as covert exfiltration channels. More recently, Guri et al proposed using the low-frequency magnetic fields emitted from the computer CPU for covert data exfiltration in order to leak

TABLE I: Summary of existing air-gap covert channels

| Type | Method |
|----------------------|--|
| Electromagnetic | AirHopper [10], [2] (FM radio) GSMem [14] (cellular frequencies) USBee [36] (USB bus emission) Funthenna [35] (GPIO emission) PowerHammer (power lines) [39] |
| Magnetic | MAGNETO [38] (CPU-generated magnetic fields) ODINI [37] (Faraday shields bypass) |
| Acoustic | Ultrasonic (speaker-to-mic) [18], [45] MOSQUITO [21] (speaker-to-speaker) Fansmitter [46] (fans noise) DiskFiltration [47] (hard disk noise) |
| Thermal | BitWhisper [17] (heat emission) |
| Optical | LED-it-GO [16] (hard drive LED) VisiSploit [43] (invisible pixels) Fast blinking images [44] Keyboard LEDs [15] CTRL-ALT-LED: keyboard LEDs [40] Router LEDs [41] air-Jumper [42] (Infrared, security cameras) |
| Vibrations (Seismic) | AiR-ViBeR [48] (Surface vibrations) |

data from Faraday caged air-gapped computers [37], [38]. In 2018, Guri et al also introduced PowerHammer, a method in which malware on air-gapped computers exfiltrates data through the main power lines using conducted emissions [39]. Several studies have proposed the use of optical emanation for air-gap communication. In 2002, Loughry and Umphress proposed the exfiltration of data by blinking the LEDs on the PC keyboard [15]. in 2019, Guri et al presented CTRL-ALT-LED, a malware which can exfiltrate data from an air-gapped computer via the keyboard LEDs of modern USB keyboards [40]. In 2016, Shamir et al demonstrated how to establish a covert channel through the air-gap using a malware, remote lasers, and scanners [23]. Guri et al also presented covert channels that use the hard drive indicator LED [16], the router LEDs [41], and security camera IR LEDs [42] to leak data from air-gapped networks. VisiSploit [43] is another optical covert channel in which data is leaked through a hidden image projected on an LCD screen. Guri also showed how to exfiltrate data from air-gapped computers via fast blinking images [44]. BitWhisper [17] is a thermal based covert channel which enables covert communication between two adjacent air-gapped computers via the exchange of so-called 'thermal-pings.'

In acoustic covert channels, data is transmitted via audible or inaudible sound waves. In 2005, Madhavapeddy et al [20] discuss 'audio networking,' which allows data transmission between a pair of desktop computers, using off-the-shelf speakers and a microphone. In 2013, Hanspach and Goetz [18] extended this method for near-ultrasonic covert networking between

air-gapped laptops using built-in speakers and microphones. They created a mesh network and used it to implement an air-gapped key-logger which demonstrates the covert channel. The concept of communicating over inaudible sounds has been extended for different scenarios using laptops and smartphones [19]. In 2018, researchers presented MOSQUITO [21], a covert communication channel between two air-gapped computers (without microphones) via so-called 'speaker-to-speaker' communication. In 2020, researcher presented a new type of vibrational (seismic) covert channel dubbed AiR-ViBeR. In this technique, data is modulated in unnoticeable vibrations generated by the computer fans. The vibrations can be received and decoded by nearby smartphones via the integrated, sensitive accelerometers.

A. Speaker-less Methods

All of the acoustic methods described above require the presence of external or internal speakers in the transmitting computer. This is considered a restrictive requirement, since loudspeakers are commonly forbidden in air-gapped computers [24], [22]. The elimination of loudspeakers is the most effective defense against the speaker-to-microphone and speaker-to-speaker covert channels discussed above [49].

In 2016, Guri et al presented DiskFiltration, a method that uses the acoustic signals emitted from the hard disk drive (HDD) to exfiltrate data from air-gapped computers [47]. Although this method doesn't need a speaker, it is limited in terms of distance (up to two meters), and it doesn't work on newer technologies such as solid-state drives (SSDs). In 2016, Guri et al also introduced Fansmitter, malware which facilitates the exfiltration of data from an air-gapped computer via noise intentionally emitted from PC fans [46]. In this method, the transmitting computer does not need to be equipped with audio hardware or an internal or external speaker. This method uses the internal fans, that exist in most laptops, desktops and server computers, and it is effective for longer distances of 6-7 meters and more. POWER-SUPPLaY differs from DiskFiltration and Fansmitter methods in two aspects. First, it uses only basic CPU instructions and does not use system resources like the HDD (DiskFiltration) or fans (Fansmitter). This makes it difficult for detection systems to identify the malicious activity of the transmitter. Second, POWER-SUPPLaY can operate at bit rates of 50 bit/sec which is much faster than Fansmitter (1 bit/sec) and DiskFiltration (3 bit/sec). In addition, POWER-SUPPLaY has the flexibility of generating acoustic tones in the 0-24 kHz frequency band, which implies that it is capable of generating both audible and inaudible sounds. Figure 3 shows the spectrogram of a part of the song 'Happy Birthday' as played form the power supply using the POWER-SUPPLaY technique.

Table 1 summarizes the existing covert channels for air-gapped computers.

IV. TECHNICAL BACKGROUND

In this section we provide the technical background on switch-mode power supplies (SMPSs), discuss the acoustic

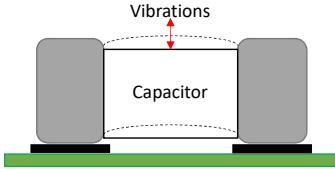


Fig. 4: Illustration of the *singing capacitor* phenomenon

emission of SMPSs and describe the signal generation.

A. Switch-Mode Power Supplies

Computers consume power using their power supplies. Modern SMPS are used in all types of electronic equipment today, including computers, TVs, printers, Internet of Things (IoT) and embedded devices, and cell phone chargers. The advantages of SMPSs over the older linear power supplies include their higher efficiency, smaller size, and lighter weight. An in-depth discussion on the design of SMPSs is beyond the scope of this paper, and we refer the interested reader to handbooks on this topic [50]. Briefly, an SMPS transfers power from a 220V AC source to several DC loads while converting voltage and current characteristics. In the SMPS, the AC power passes through fuses and a line filter, and is then rectified by a full-wave bridge rectifier. The rectified voltage is applied to the power factor correction module and regulated via DC to DC converters.

B. SMPS Acoustic Emission

The DC supply from a rectifier or battery is fed to the inverter where it is turned on and off at high rates by the switching MOSFET or power transistors. This switching rate is also known as the *switching frequency* of the SMPS. The typical switching frequency of a computer power supply is between 20 kHz and 20 MHz.

The switching frequency affects, among others components in the SMPS, the transformers and capacitors. These are the primary sources of the acoustic noise generated by the SMPS.

- Transformers' Audible Noise.** Transformers produce audible noise, since they contain many physically movable elements, such as coils, isolation tapes, and bobbins. The current in the coils, which occurs at the switching frequency, produces electromagnetic fields which generate repulsive and/or attractive forces between the coils. This can produce a mechanical vibration in the coils, ferrite cores, or isolation tapes.

- Capacitors Audible Noise.** Ceramic capacitors can produce audible noise, since they have Piezoelectric characteristics. The Piezoelectric acoustic effect on the capacitor is commonly described as "singing capacitors". This noise is actually the result of vibrations of the capacitor on the Printed Circuit Board (PCB) that occur in normal working conditions. These vibrations causes capacitor displacement, as shown in Figure 4. The frequency and amplitude of the displacement determine the acoustic waveform generated from the capacitors. When the vibration frequency occurs within the audible range,

approximately 20 Hz – 20 kHz, it may also be heard as an audible hum. The range between 20 kHz and 24 kHz is considered as 'near-ultrasonic' and can not be heard by most humans.

The typical switching frequency of SMPS during its normal operation is within the range of 20kHz-20MHz. Thus, the acoustic signal generated by SMPS is mainly in a frequency range of 20kHz and higher. This range is at the upper bound of human hearing and considered inaudible to adult humans.

V. TRANSMISSION

A. Signal Generation

Since modern CPUs are energy efficient, the momentary workload of the CPU directly affects the dynamic changes in its power consumption [51]. By regulating the workload of the CPU, it is possible to govern its power consumption, and hence control the momentary *switching frequency* of the SMPS. By intentionally starting and stopping the CPU workload, we are able to set the SMPS so it switches at a specified frequency and hence emit an acoustic signal and modulate binary data over it.

To generate a switching frequency f_c , we control the utilization of the CPU at a frequency correlated to f_c . To that end, n worker threads are created where each thread is bound to a specific core. To generate the carrier wave, each worker thread overloads its core at a frequency f_c repeatedly alternating between applying a continuous workload on its core for a time period of $1/2f_c$ and putting its core in an idle state for a time period of $1/2f_c$.

Algorithm 1 transmit (cores, freq, time)

```

1: pthread_barrier_init(&halfCycleBarrierHI, NULL, cores)
2: pthread_barrier_init(&halfCycleBarrierLO, NULL, cores)
3: for i ← 1 to cores do
4:   threadCreate(worker)
5: end for
6: end = clock_gettime() + time
7: halfCycleNano ← NANO_PER_SECOND/freq * 0.5
8: cycleNano ← NANO_PER_SECOND/freq
9: while clock_gettime() < end do
10:   LO ← 0
11:   pthread_barrier_wait(&halfCycleBarrierLO)
12:   while clock_gettime()%cycleNano < halfCycleNano do
13:     end while
14:   LO ← 1
15:   pthread_barrier_wait(&halfCycleBarrierHI)
16:   while clock_gettime()%cycleNano >= halfCycleNano do
17:     end while
18: end while

```

Algorithms 1 shows the generation of acoustic tone (*freq*) for a time duration of (*time*) milliseconds, using (*nCores*)

Algorithm 2 worker (threadState)

```

1: while true do
2:   //sync threads on end of LO half cycle
3:   pthread_barrier_wait(&halfCycleBarrierLO)
4:   //HI half cycle – busy loop
5:   while !LO do
6:   end while
7:   //sync threads on end of HI half cycle
8:   pthread_barrier_wait(&halfCycleBarrierHI)
9: end while

```

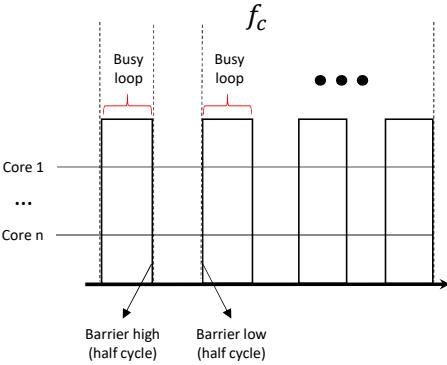


Fig. 5: n cores utilization during a transmission

cores. At the beginning we initiate ($nCores$) threads and bind each thread to a specific core. The switching frequency is generated by each worker thread by employing busy loops and barrier objects. We overload the core using the busy waiting technique. This causes full utilization of the core for the time period and returns. The worker threads are synchronized with barrier objects, allowing them to start and stop the switching frequency altogether. The main thread governs the synchronization of the worker threads, changing the cycle state flag according to the barrier timing. This ensures that the generation of the duty cycles are precisely timed between the cores. Figure 5 illustrates the cores' utilization during the generation of a carrier wave f_c . Note that the power of the generated signal is correlated to the number of cores that participate in the transmission. The utilization of more cores yields more power consumption and hence more 'dancing capacitors'.

Based on the algorithm above, we implemented a transmitter for Linux Ubuntu (version 16.04, 64 bit). We used the `sched_setaffinity` system call to bind each thread to a CPU core. The affinity is the thread level attribute that is configured independently for each worker thread. To synchronize the initiation and termination of the worker threads, we used the thread barrier objects with `pthread_barrier_wait()` [52]. Note that the precision of `sleep()` is not sufficient for our needs given the frequencies of the carrier waves which are at 24kHz or lower.

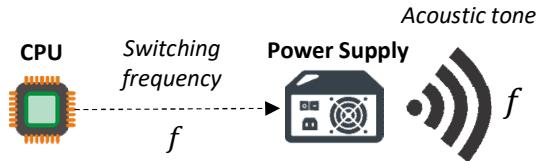


Fig. 6: The signal generation

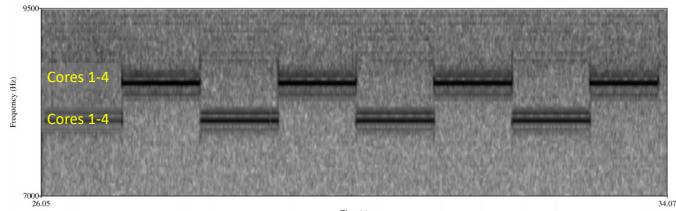


Fig. 7: Spectrogram of the FSK modulation. The acoustic signals generated from the PC power supply.

B. Data Transfer

The acoustic signal can be used to carry digital information using modulation schemes. For the data transfer we used the Frequency-Shift Keying (FSK) and the more advanced Orthogonal Frequency-Division Multiplexing (OFDM) modulation schemes.

1) Frequency-Shift Keying: In frequency-shift keying (FSK) the data is represented by a change in the frequency of a carrier wave. Recall that the transmitting code can determine the frequency of the generated signal. In FSK, each frequency represents a different symbol.

Figure 6 illustrates the generation of FSK signals. In this case, n threads are bounded to the CPU cores. All threads are transmitting in the same carrier at given time, and hence generating a distinct switching frequency for each symbol. The generated acoustic tone is correlated with this switching frequency.

Figure 7 shows the time-frequency spectrogram of a binary sequence ('010101010') modulated with two frequency FSK (B-FSK) as transmitted from a PC with four cores. In this modulation, the frequencies 8500Hz and 8750Hz have been used to encode the symbols '0', '1', respectively.

2) Orthogonal Frequency-Division Multiplexing: We developed a fine-grained approach, in which we control the workload of each of the CPU cores independently from the other cores. Regulating the workload of each core separately enables greater control of the momentary switching frequency. By controlling the workload of each core separately, we can use a different sub-carrier for each transmitting core. This allows us to employ a more efficient modulation scheme such as orthogonal frequency-division multiplexing (OFDM).

In orthogonal frequency-division multiplexing data is represented by multiple carrier frequencies in parallel. In our case, we use different cores to transmit data in different sub-carriers. In each sub-carrier, we used on-off keying (OOK) to modulate the data. Note that since the sub-carriers' signals are generated

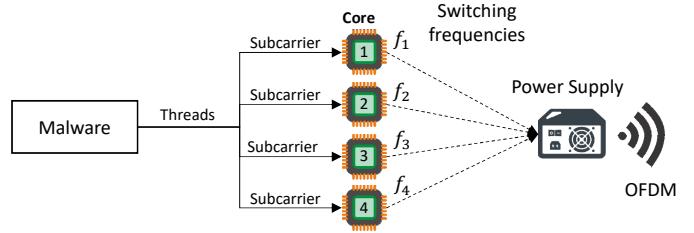


Fig. 8: Illustration of the OFDM modulation

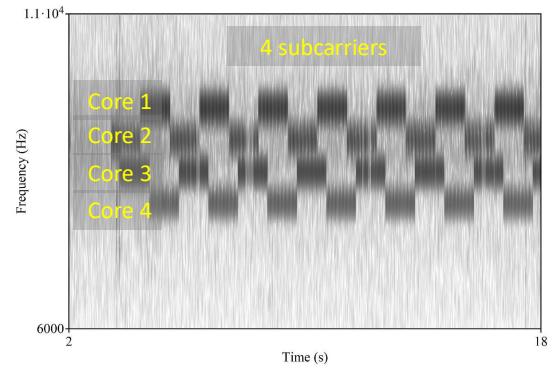


Fig. 9: Spectrogram of the OFDM modulation

in parallel, the maximal number of sub-carriers is equal to the number of cores available for the transmissions.

Figure 8 illustrates the generation of OFDM signals. In this case, 4 threads are bounded to the 4 CPU cores. Each thread transmits in different sub-carrier, and hence generates different switching frequency. The acoustic tone is compound from all the switching frequencies generated by the sub-carriers.

Figure 9 presents a binary sequence modulated with OFDM with four sub-carriers as transmitted from a PC with four cores. In this modulation, 8000Hz (core 1), 8200Hz (core 2), 8400Hz (core 3) and 8600Hz (core 4) have been used to encode the symbols '00', '01', '10' and '11', respectively.

C. Transmission Protocol

We transmit the data in small packets composed of a preamble, a payload, and a cyclic redundancy check (CRC) code.

- Preamble.** A preamble header is transmitted at the beginning of every packet. It consists of a sequence of eight alternating bits ('10101010') which helps the receiver determine the carrier wave frequency and amplitude. In addition, the preamble allows the receiver to detect the beginning of a transmission. Note that in our covert channel the amplitude of the carrier wave is unknown to the receiver in advance, and it mainly depends on what type of transmitting computer is used, the number of cores participating in the transmission, and the distance between the transmitter and the receiver. These parameters are synchronized with the receiver during the preamble.

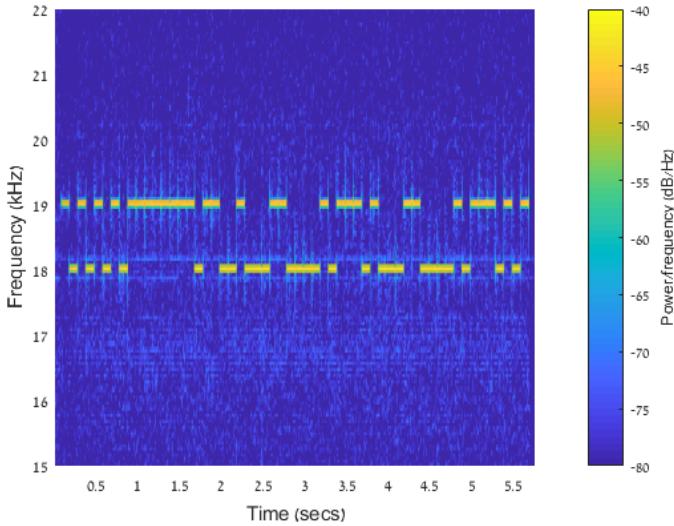


Fig. 10: Spectrogram of a frame modulated with B-FSK as transmitted from a PC power supply

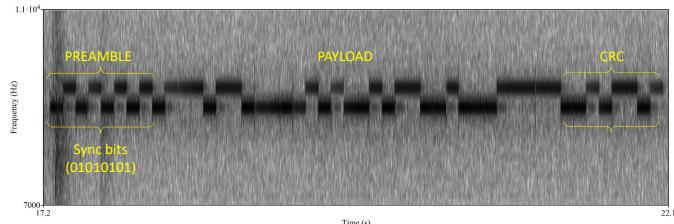


Fig. 11: Spectrogram of a frame modulated with B-FSK as transmitted from a PC power supply

- **Payload.** The payload is the raw data to be transmitted. In our case, we arbitrarily choose 32 bits as the payload size.
- **CRC.** For error detection, an eight bit CRC (cyclic redundancy check) is added to the end of the frame. The receiver calculates the CRC for the received payload, and if it differs from the received CRC bit, an error is detected. A spectrogram of a full frame transmitted from a computer is presented in Figure 11. In this case, a B-FSK was used with frequencies of 8400Hz and 8600Hz.

VI. AUDIO FORMATS

We implemented a program which plays digital audio streams through the power supply. Our implementation supports the Waveform Audio File (WAV) format. The most common WAV format encodes uncompressed audio in pulse code modulation (PCM). There are two fundamental parameters which should be considered when playing audio streams: the *sample rate* and the *bit depth*. We briefly discuss each with regard to playing audio through the power supply.

A. Sample Rate

The sampling rate (measured in kHz) of an audio file is the number of samples of audio carried per second. According to the Nyquist Theorem, the maximum frequency that can

be represented in an audio file is half of the sample rate. Most sound cards can play audio streams at 48kHz, which is the sample rate used for DVDs. In the case of SMPS, the maximum playing frequency depends on the maximum rate at which we can start and stop utilizing the CPU cores, which directly affects the SMPS switching frequency. Using the barriers and busy loop technique described above, we could switch each CPU core at 100kHz, and hence are capable of playing audio streams at lower rates of 48kHz.

B. Bit Depth

In digital audio the bit-depth represents the number of possible amplitude values for a sample. The amplitude of each sample is encoded by the number of bits, which is the bit depth of the audio stream. In standard sound cards, the amplitude of a sample is sent to the digital-to-analog converter (DAC) component and played through the loudspeakers. The loudspeakers' membranes vibrate in a power correlated to the amplitude that produces the desired signal. In the case of SMPS, the amplitude of the signal generated from the capacitors is constant and produces square acoustic waves. This represents a simple audio stream with a depth of only 1-bit (signal/no-signal). In order to play complex audio streams with a wider bit depth, we implemented two different modulation techniques; (1) amplitude modulation (AM), and (2) pulse width modulation (PWM).

1) *Amplitude Modulation:* As described in the signal generation section, the number of cores used for signal generation affects the strength of the signal. This is because the more cores that are utilized, the more capacitors and transformers that are switched in the SMPS. This enables some control of the *amplitude* generated for each sample. Given N virtual cores, it is possible to play each sample at N different amplitudes and hence play audio streams with a bit depth of roughly N . Figure 12 shows the waveform of a signal with three amplitudes generated with the AM technique. In this case, the signal is generated using four, two and one cores.

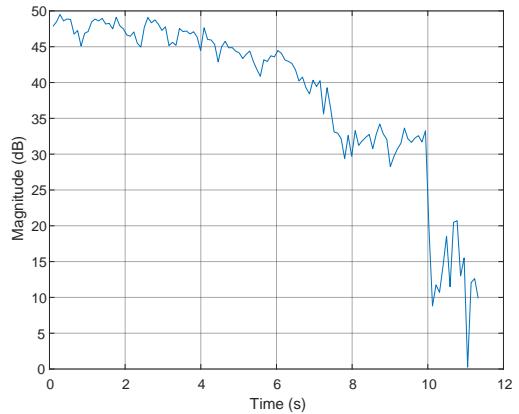


Fig. 12: Three amplitudes generated with the AM technique

2) *Pulse Width Modulation:* PWM is a method of reducing the average power delivered by an electrical signal, by effectively breaking it into discrete parts. The average value

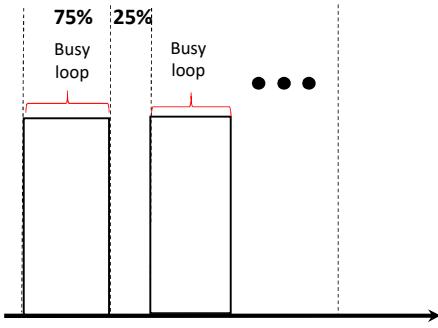


Fig. 13: Illustration of a PWM modulated signal

of voltage (and current) fed to the load is controlled by turning the switch between the supply and load on and off at a fast rate. The longer the switch is on compared to the time it is off, the greater the total amount of power supplied to the load.

By carefully timing a short pulse (i.e., going from one output level to the other and back), the end result corresponds to intermediate sound levels, functioning as a crude DAC. PWM allows an approximate playback of WAV and PCM audio.

The sound level of a sample is determined by the duration of the CPU utilization. This is the *duty cycle* of the signal. A short duty cycle corresponds to low power (low volume), because the power is off for most of the time, while a long duty cycle corresponds to high power (high volume). We calculate the duty cycle required for each sample given its amplitude value. Figure 13 illustrates a signal generated with a duty cycle of 75% which implies a signal which is 25% stronger than an average signal in the stream. Figure 14 contains the spectrogram of a signal with eight decreasing amplitudes, generated with the PWM technique. In this case, the signal is generated using eight duty cycles which decrease over time.

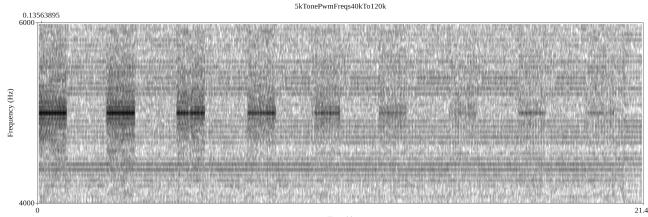


Fig. 14: Eight levels of sound using the PWM technique

C. Play WAV via Power-Supply

Based on the AM and PWM techniques described above, we implemented a program which plays PCM and WAV files through the power supply. Our `PowerSupply [AM | PWM] wav_file.wav` program determines which type of modulation to use and plays the `wav_file` using the corresponding technique.

VII. RECEPTION

We implemented a receiver as an app for the Android OS. The main functionality of the receiver is (1) sampling

the signal, (2) performing signal processing, (3) detecting the preamble header, (4) demodulating the payload, and (5) handling errors.

The main functionality of the receiver is run in a separate thread. It is responsible for data sampling, signal processing, and data demodulation. An outline of the receiver is presented in Algorithm 1; this is followed by a description.

Algorithm 1

```

1: procedure RECEIVER
2:   raw_signal = sample(44.1kHz)
3:   signal = FFT(raw_signal)
4:   if (state == PREAMBLE) then
5:     if (DetectPreamble(signal) == true) then
6:       T, f0, f1 = ExtractChannelParams(signal)
7:       SetState(DEMODULATE)
8:     end if
9:   end if
10:  if (state == DEMODULATE) then
11:    bit = Demodulate(signal)
12:    data.add(bit)
13:    if (data.size%(32 + 8) == 0) then
14:      SetState(PREAMBLE)
15:    end if
16:    if (SignalLost(signal)) then
17:      SetState(PREAMBLE)
18:    end if
19:  end if
20: end procedure

```

1) *Signal Sampling and Downsampling:* In order to analyze the waveforms in the frequency domain, we record data at a rate of 44.1kHz and save it to a temporary buffer.

2) *Signal Processing:* The first step is to measure the signal and transfer it to the frequency domain. This step includes performing a fast Fourier transform (FFT) on the sampled signal. The signal measured is stored in a buffer after applying a filter for noise mitigation. This data is used later in the demodulation routines. The noise mitigation function is applied to the current sample by averaging it with the last w samples.

3) *Preamble Detection:* In the *PREAMBLE* state the receiver searches for a preamble sequence to identify a frame header (lines 5-8). If the preamble sequence '10101010' is detected, the state is changed to *DEMODULATE* to initiate the demodulation process. Based on the preamble sequence, the receiver determines the channel parameters, signal time, (T) and frequencies f_0 and f_1 .

4) *Demodulation:* In the *DEMODULATE* state the payload is demodulated given the signal parameters retrieved in the *PREAMBLE* state. The demodulated bit is added to the current payload. When the payload of 32 bits and the 8-bit CRC are received, the algorithm returns back to the preamble detection state (*PREAMBLE*).

5) *Error Handling:* The `SignalLost()` function returns true if, during the data reception, the signal power measured in

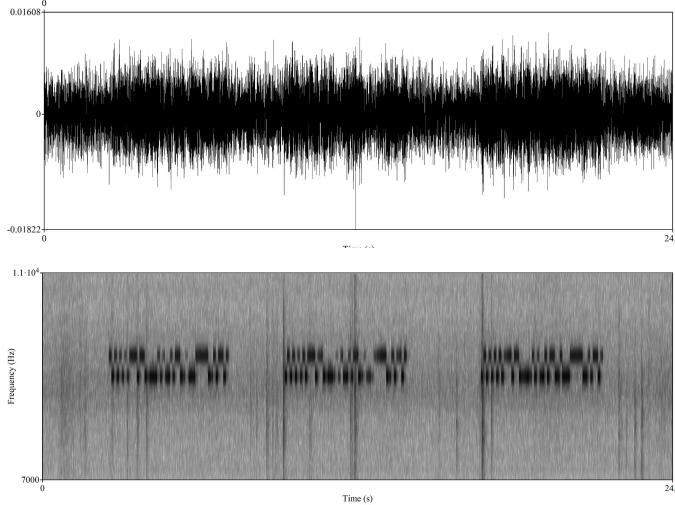


Fig. 15: The raw signal and the demodulated bits

the carrier frequency is weaker than the amplitude of the '0's from the preamble for three seconds straight. In this case, any partially received data is discarded, and the function returns to the *PREAMBLE* state. Signal loss may occur if the malware stops the transmission (e.g., for stealth or due to the computer shutting down). Signal loss may also occur if the receiving smartphone has been moved away from the transmitting computer.

The demodulation is depicted in Figure 15. The upper image presents the raw acoustic signal recorded by a nearby smartphone. The spectrogram in the lower image shows the demodulated bits.

VIII. EVALUATION

In this section we provide an analysis and evaluation of the acoustic waveforms generated through the power supply. We also present the experimental setup and discuss the measurements and results.

A. Measurements Setup

1) Transmitters: We examined the acoustic signal generated from six types of computers: three standard desktop workstations (PC-1, PC-2, and PC-3), a server machine with multi-core processors (Server), a small form factor computer (NUK), and a low power embedded device (IOT). A detailed list of the computers tested and their technical specifications is provided in Table II.

2) Receiver: During the tests, the acoustic signals were recorded with a Samsung Galaxy S7 (G930F) smartphone, running the Android OS. The audio sampling was performed using the smartphone's internal microphone at a sampling rate of 44.1kHz. We used the *MathWorks MATLAB* environment for signal processing, and the *Praat* framework for additional spectral analysis [53].

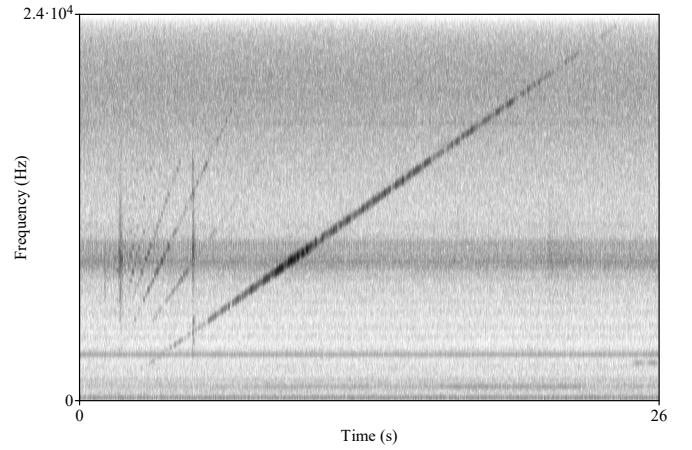


Fig. 16: The sweep signal recorded from PC-1 (spectrogram view)

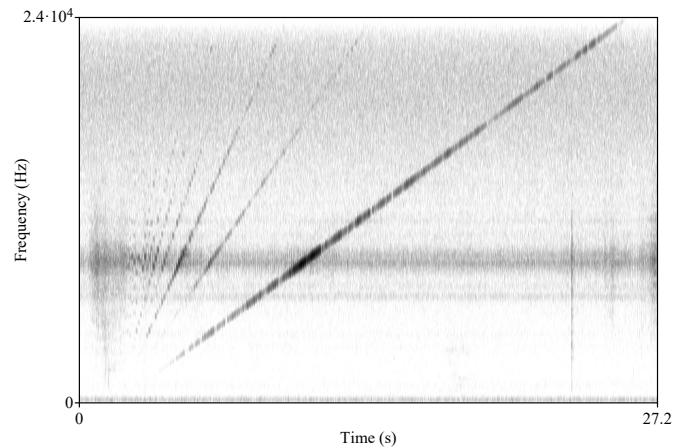


Fig. 17: The sweep signal recorded from NUK (spectrogram view)

B. Spectral View

Power supplies are not intended to generate sound and hence are limited in terms of the acoustic signal they can produce. Unlike loudspeakers which can generate sound waves in the entire spectrum of 0-24kHz, a good quality signal generated from the PSU transformers and capacitors is only obtained at certain fragmented bands of the spectrum. These ranges are largely specific to each PSU, and dependent on their internal design, structure, and the type of transformers and capacitors.

We examined the frequency bands of each of the six PSUs by analyzing the *chirp signal* they generate. This signal, also known as a *sweep signal*, is generated by gradually increasing the switching frequencies of the PSU from the minimum to the maximum in a specific range. In our case, we examined the sweep signal in the band of 0-24kHz which can be received by a nearby smartphone. For the analysis of the chirp signal, the recording smartphone was located 20 cm away from the tested device. Table III summarizes the analysis of the sweep signals recorded from the six computers. It presents the frequency

TABLE II: The computers used in the experiments

| # | Type | Model | PSU | CPU |
|--------|-------------------|--------------------|---------------------------|---|
| PC-1 | Desktop PC | Silverstone | FSP300-50HMN 300W | Intel Core i7-4770 CPU@ 3.4GHz (4 cores) |
| PC-2 | Desktop PC | Infinity | PK-450 450W | Intel Core i7-4770 CPU 3.4GHz (4 cores) |
| PC-3 | Desktop PC | Lenovo | ATX POWER SUPPLY 280 WATT | Intel Core Quad-Q9550 CPU @2.83GHz (4 cores) |
| Server | Server | IBM | DPS-750AB A 750Wx2 | Intel Xeon CPUE5-2620 12 cores (24 threads) |
| NUK | Small form factor | Lenovo ThinkCentre | PA-1650-72 20V 3.25A | Intel Core i7 -4785T (4 cores) |
| IOT | Embedded/IoT | Raspberry Pi 3 | DSA-13 PFC-05, 5V 2.5A | Quad Core Broadcom BCM2837, 64-bit, ARMv8, processor Cortex A53 |

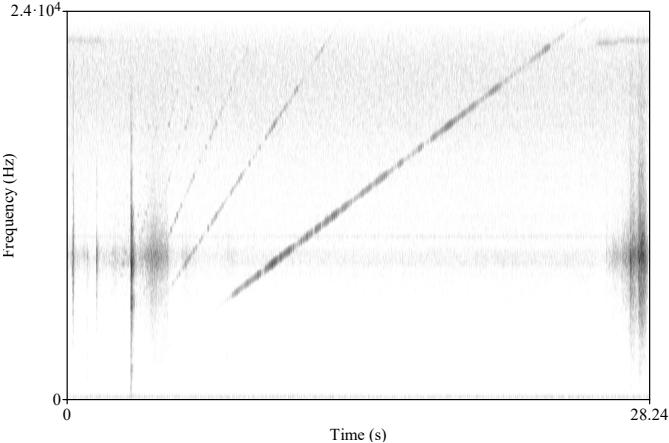


Fig. 18: The sweep signal recorded from IOT (spectrogram view)

band generated from each computer and an estimation of the signal quality based on the SNR ranges measured from the entire sweep signal.

TABLE III: The frequencies of the audible bands of for six PSUs under test

| # | Best audible band | Signal quality (SNR range) |
|--------|----------------------------------|----------------------------|
| PC-1 | 2300Hz - 22000Hz | High (30-35dB) |
| PC-2 | 2200Hz - 22000Hz | High (30-35dB) |
| PC-3 | 5491Hz - 6223Hz, 8300Hz - 9000HZ | Low (20-30dB) |
| Server | 8000Hz - 17000Hz | Low (10-20dB) |
| NUK | 1800Hz - 24000Hz | High (30-40dB) |
| IOT | 3100Hz - 24000Hz | Intermediate (20-30dB) |

The results show that the quality of the acoustic signal may vary among different types of computers. PC-1 and PC-2 generated a detectable signal in a band of 2300Hz-22000Hz, while PC-3 could only generate a weak signal on the fragmented bands of approximately 5500Hz-6220Hz and 8300Hz-9000Hz. The server computer generated a weak signal in a band of 8000Hz-17000Hz. The NUK computer generated a strong signal in the entire band of 1800Hz-24000Hz, and the IOT generated a intermediate signal in a band of 3100Hz-24000Hz. It is important to note that the strength of the signals across the spectrum are not uniform; the signal may be strong in certain frequency bands and weak in others. Based on the results, we chose PC-1, NUK, and IOT for the reminder of the evaluation. These computers produce strong signals, and they represent the common cases of a PC workstation, small form

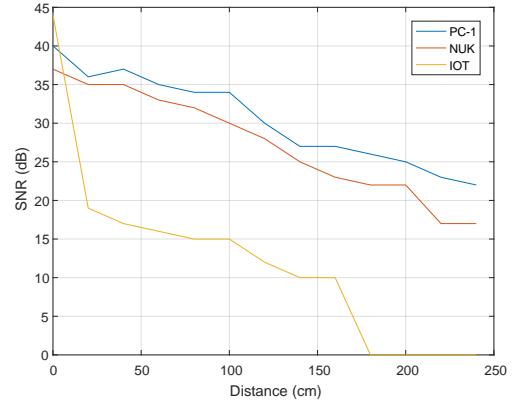


Fig. 19: SNR levels measured at distances of 0-240cm

factor computer, and a low-power IoT device. The spectrogram of their chirp signals can be seen in Figures 16, 17, and 18, respectively.

C. Signal-to-Noise Ratio (SNR)

The acoustic signals generated from the PSU transformers and capacitors are limited in terms of strength. We measured the signal-to-noise ratio (SNR) to compare the level of a generated signal (S) to the level of the background noise (N) at different distances. This measure is correlated to the quality of the audible signal, e.g., an S/N ratio greater than one indicates there is more signal than noise. We generated a signal for a time period of one second and compared it with the background noise. For the SNR measurements, we used the frequency band with the strongest signal for each PSU. The signals were recorded with the smartphone receiver located at distances of 0-250cm from the computer. The results are presented in Figure 19.

As can be seen, the acoustic signal emanating from the IOT device could be received from a distance of about 150cm with an SNR of 10dB. With PC-1 and NUK, the good quality signals could be received from a distance of 250cm with 22dB and 17dB, respectively.

D. Number of Cores

As discussed, the number of cores used for signal generation directly influences the strength (amplitude) of the signal (i.e., more transmitting threads yield a stronger signal). Figure 20 presents the SNR measurements of PC-1 with one, two, and four cores used for signal generation. In this test, we used one thread per core. The signal recorded at a distance of 20cm

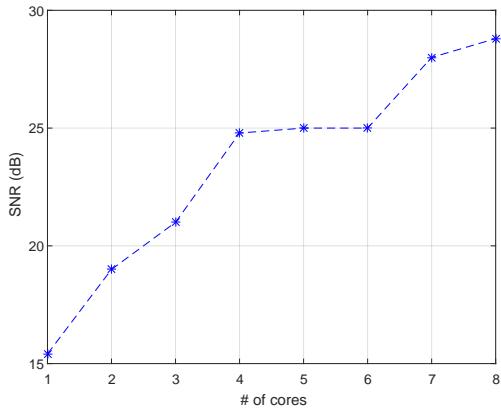


Fig. 20: SNR levels measured with 1 to 8 cores

from the computers showed a gradual increase as the number of cores used for the signal generation increased (this ranged from an SNR of 15dB with one core to an SNR of 27dB with eight cores).

a) Hyper-Threading: Note that modern Intel CPUs support hyper-threading technology [54]. With this technology, each physical core exposes two logical (virtual) cores to the operating system. The CPU shares the workload between the logical cores when possible for better utilization. In the experiments, we bound the transmitting threads to the systems' logical cores rather than the physical cores, i.e., in a system with four physical cores and eight virtual cores we can potentially run eight concurrent transmitting threads.

E. Bit-Error-Rate (BER)

As discussed in Section V-B, the acoustic signal can be used for data transfer. We measured the bit error rate for PC-1, NUK, and IOT at distances of 0-240cm for PC-1 and NUK and 100cm for the IOT device. For the tests, we used B-FSK modulation and the frequency band with the strongest signal for each PSU. We transmitted sequences of random bits, decoded them, and compared the results with the original data. The tests were repeated three times. The results for PC-1, NUK, and IOT are summarized in Tables IV, V, and VI, respectively, as described below.

1) PC-1: With bit rates of 1 bit/sec to 10 bit/sec we could maintain data transmission at distances of 240cm with a maximum BER of 5%. At a rate of 30 bit/sec we could maintain data transmission at distances of 100cm with a maximum BER of 10%. With a bit rate of 60 bit/sec we could maintain data transmission at distances of 60cm with a maximum BER of 0%.

2) NUK: With bit rates of 1 bit/sec to 20 bit/sec we could maintain data transmission at distances of 220cm with a maximum BER of 5% and distances of 240cm with a maximum BER of 8%. With a bit rate of 50 bit/sec we could maintain data transmissions at distances of 100cm with a maximum BER of 10%. With a bit rate of 60 bit/sec we could



Fig. 21: The distances and relevant data for exfiltration

maintain data transmission at short distances of 20cm with a maximum BER of 0%.

3) IOT: With bit rates of 1 bit/sec, 20 bit/sec, and 40 bit/sec, we could maintain data transmission at distances of 100cm with a maximum BER of 2-3%. With a bit rate of 80 bit/sec, we could maintain data transmission at distances of 20cm with a maximum BER of 10%.

F. Greater Distance

We found that some power supplies produce strong acoustic signals in certain frequency bands. We were able to receive acoustic signals up to six meters away from PC-4. However, the quality of the signal significantly decreases with the distance. From 4m, 5m and 6m away we were able to receive the signals at maximal frequencies of 14kHz, 12kHz and 9kHz, respectively. Table VII shows the results of PC-4. With bit rates of 5 bit/sec we could maintain data transmission at distances of 6m away with a maximum BER of 1.2%.

G. Threat Radius

The above results indicates that the covert channel can be used to transfer data with bit rates of 0-40 bit/sec at short distances of 0-100cm. Such bit rates could be used to transmit binary data, keystrokes logging, text files, and so on. With slower bit rates of 1-10 bit/sec, the covert channel is effective for even greater distances (200-300cm with PC-1 and NUK, 200cm-600cm with PC-4). The slower bit rates could be used to transfer a small amount of data, such as short texts, encryption keys, passwords, and keystroke logging, as seen in Figure 21.

H. Stealth

The transmitting code shown above requires no elevated privileges and can be initiated from an ordinary user space process. The code consists of basic CPU operations such as busy loops, which do not expose malicious behavior, making it highly evasive and unable to detect by static and dynamic (runtime) malware detection solutions.

IX. COUNTERMEASURES

There are four main categories of countermeasures that can be used to defend against the proposed covert channel: zones separation, signal detection, signal jamming, and signal blocking.

TABLE IV: Bit error rates (BER) for PC-1

| # | bps | 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 | 220 | 240 |
|------|-----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|
| PC-1 | 1 | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 5% |
| PC-1 | 10 | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 5% |
| PC-1 | 30 | 0% | 0% | 0% | 0% | 5% | 10% | - | - | - | - | - | - | - |
| PC-1 | 60 | 0% | 0% | 0% | - | - | - | - | - | - | - | - | - | - |

TABLE V: Bit error rates (BER) for NUK

| # | bps | 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 | 220 | 240 |
|-----|-----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|
| NUK | 1 | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 5% | 8% |
| NUK | 20 | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 5% | - |
| NUK | 50 | 0% | 0% | 0% | 0% | 5% | 10% | - | - | - | - | - | - | - |
| NUK | 60 | 0% | 0% | - | - | - | - | - | - | - | - | - | - | - |

TABLE VI: Bit error rates (BER) for IOT

| # | bps | 0 | 20 | 40 | 60 | 80 | 100 |
|-----|-----|----|-----|----|----|----|-----|
| IOT | 1 | 0% | 0% | 0% | 0% | 0% | 2% |
| IOT | 20 | 0% | 0% | 0% | 0% | 0% | 3% |
| IOT | 40 | 0% | 0% | 0% | 0% | 3% | 3% |
| IOT | 80 | 0% | 10% | - | - | - | - |

TABLE VII: Bit error rates (BER) for PC-4 with 5 bps

| # | 1m | 2m | 3m | 4m | 5m | 6m |
|------|----|----|------|------|------|------|
| PC-4 | 0% | 0% | 1.1% | 1.2% | 1.2% | 1.2% |

a) *Zoning*: In procedural countermeasures the 'zoning' approach may be used. In this approach sensitive computers are kept in restricted areas in which mobile phones, microphones, and electronic equipment are banned. The zoning approach (also referred to as 'red/black separation') is discussed in [55] as a means of handling various types of acoustic, electromagnetic, and optical threats. However, zoning is not always possible, due to practical limitations such as space and cost. In our case, recording devices such as mobile phones should be banned from the area of air-gapped systems or be kept at a certain distance from them.

b) *Signal Detection*: Host based intrusion detection systems (HIDS) may continuously trace the activities of running processes in order to detect suspicious behavior; in our case, a group of threads that abnormally regulates the switching frequency would be reported and inspected. Such a detection approach would likely suffer from false alarms, since many legitimate processes use CPU intensive calculations that affect the processor's workload [56]. Another problem in the runtime detection approach is that the signal generation algorithm (presented in Section V) involves only non-privileged CPU instructions (e.g., busy loops). Monitoring non-privileged CPU instructions at runtime necessitates that entering the monitored processes enter a *step-by-step* mode, which severely degrades performance [14]. Software based detection also suffers from an inherent weakness in that it can easily be bypassed by malware using evasion techniques [57]. In our case, the malware may inject the transmitting threads into a legitimate, trusted process to bypass the security mechanisms.

Hardware-based countermeasures may include noise detector devices which monitor the spectrum at a range of

frequencies. Such products exist [58] but are prone to false alarms due to natural environmental noises [56]. Jamming the PSU signal by the generating background noise at a specific ranges [59] is also possible but not applicable in some environments, particularly in quiet settings. In our case, the frequency band is 0-24kHz which is include the audible region.

c) *Signal Jamming*: Jamming the whole frequency band will generate a noticeable amount of environmental noise which may disturb users. Physical isolation in which the computer chassis is built with special noise blocking cover is also an option, but it is costly and impractical on a large-scale. Carrara [45] suggested monitoring the audio channel for abnormally energy peaks, in order to detect hidden transmissions in the area. In our case, the ultrasonic frequency range above 18kHz should be scanned (continuously) and analyzed. However, as noted in [45], if the monitoring device is far from the transmitter this approach may not be effective.

d) *Signal Limiting/Blocking*: Although there are quiet PSUs that limit the acoustic noise emitted from their internal components, this feature does not *hermetically* prevent the emission of noise [50]. Physical isolation in which the computer chassis is enclosed within a special noise blocking cover is also an option, but it is costly and impractical on a large scale.

X. CONCLUSION

In this paper, we show that malware running on a computer can use the power supply as an out-of-band speaker. A code executed in the system can intentionally regulate the internal *switching frequency* of the power supply, and hence control the waveform generated from its capacitors and transformers. This technique allows sonic and ultrasonic audio tones to be generated from a various types of computers and devices even when audio hardware is blocked, disabled, or not present. We show that the POWER-SUPPLaY code can operate from an ordinary user-mode process and doesn't need hardware access or root-privileges. This proposed method doesn't invoke special system calls or access hardware resources, and hence is highly evasive. We present the implementation details and evaluation, and provide the measurement results. By using POWER-SUPPLaY, we could acoustically exfiltrated data from audio-less systems to a nearby mobile phone at a distance of 2.5 meters with a maximal bit rate of 50 bit/sec.

REFERENCES

- [1] “Classified united states website - wikipedia,” https://en.wikipedia.org/wiki/Classified_United_States_website, 2018, (Accessed on 12/03/2017).
- [2] M. Guri, M. Monitz, and Y. Elovici, “Bridging the air gap between isolated networks and mobile phones in a practical cyber-attack,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 4, p. 50, 2017.
- [3] E. Byres, “The air gap: Scada’s enduring security myth,” *Communications of the ACM*, vol. 56, no. 8, pp. 29–31, 2013.
- [4] R. Langner, “Stuxnet: Dissecting a cyberwarfare weapon,” *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.
- [5] R. Grant, “The cyber menace,” *Air Force Magazine*, vol. 92, no. 3, 2009.
- [6] K. ZAO, “Red october diplomatic cyber attacks investigation,” 2018.
- [7] “A fanny equation: “i am your father, stuxnet” - securelist,” <https://securelist.com/a-fanny-equation-i-am-your-father-stuxnet/68787/>, 2018, (Accessed on 12/03/2017).
- [8] “No big deal... kremlin hackers ‘jumped air-gapped networks’ to pwn us power utilities the register,” https://www.theregister.co.uk/2018/07/24/russia_us_energy_grid_hackers/, (Accessed on 02/12/2020).
- [9] “An indian nuclear power plant suffered a cyberattack. heres what you need to know. - the washington post,” <https://www.washingtonpost.com/politics/2019/11/04/an-indian-nuclear-power-plant-suffered-cyberattack-heres-what-you-need-know/>, (Accessed on 02/12/2020).
- [10] M. Guri, G. Kedma, A. Kachlon, and Y. Elovici, “Airhopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies,” in *Malicious and Unwanted Software: The Americas (MALWARE), 2014 9th International Conference on*. IEEE, 2014, pp. 58–67.
- [11] M. G. Kuhn and R. J. Anderson, “Soft tempest: Hidden data transmission using electromagnetic emanations.” in *Information hiding*, vol. 1525. Springer, 1998, pp. 124–142.
- [12] M. G. Kuhn, “Compromising emanations: eavesdropping risks of computer displays,” Ph.D. dissertation, University of Cambridge, 2002.
- [13] M. Vuagnoux and S. Pasini, “Compromising electromagnetic emanations of wired and wireless keyboards.” in *USENIX security symposium*, 2009, pp. 1–16.
- [14] M. Guri, A. Kachlon, O. Hasson, G. Kedma, Y. Mirsky, and Y. Elovici, “Gsmem: Data exfiltration from air-gapped computers over gsm frequencies.” in *USENIX Security Symposium*, 2015, pp. 849–864.
- [15] J. Loughry and D. A. Umphress, “Information leakage from optical emanations,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, no. 3, pp. 262–289, 2002.
- [16] M. Guri, B. Zadov, and Y. Elovici, *LED-it-GO: Leaking (A Lot of) Data from Air-Gapped Computers via the (Small) Hard Drive LED*. Cham: Springer International Publishing, 2017, pp. 161–184.
- [17] M. Guri, M. Monitz, Y. Mirski, and Y. Elovici, “Bitwhisper: Covert signaling channel between air-gapped computers using thermal manipulations,” in *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*. IEEE, 2015, pp. 276–289.
- [18] M. Hanspach and M. Goetz, “On covert acoustical mesh networks in air,” *arXiv preprint arXiv:1406.1213*, 2014.
- [19] L. Deshotels, “Inaudible sound as a covert channel in mobile devices.” in *WOOT*, 2014.
- [20] A. Madhavapeddy, R. Sharp, D. Scott, and A. Tse, “Audio networking: the forgotten wireless technology,” *IEEE Pervasive Computing*, vol. 4, no. 3, pp. 55–60, 2005.
- [21] M. Guri, Y. Solewicz, and Y. Elovici, “Mosquito: Covert ultrasonic transmissions between two air-gapped computers using speaker-to-speaker communication,” in *2018 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2018, pp. 1–8.
- [22] “Air gap computer network security - notary colorado springs,” <http://abclegaldocs.com/blog-Colorado-Notary-air-gap-computer-network-security/>, 2018, (Accessed on 06/14/2018).
- [23] M. Guri and Y. Elovici, “Bridgeware: The air-gap malware,” *Commun. ACM*, vol. 61, no. 4, pp. 74–82, Mar. 2018. [Online]. Available: <http://doi.acm.org/10.1145/3177230>
- [24] “Mind the gap: Are air-gapped systems safe from breaches? — symantec connect community,” <https://www.symantec.com/connect/blogs/mind-gap-are-air-gapped-systems-safe-breaches>, 2018, (Accessed on 06/14/2018).
- [25] “The epic turla (snake/uroburos) attacks — virus definition — kaspersky lab,” <https://www.kaspersky.com/resource-center/threats/epic-turla-snake-malware-attacks>, 2018, (Accessed on 12/03/2017).
- [26] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, N. Modadugu *et al.*, “The ghost in the browser: Analysis of web-based malware.” *HotBots*, vol. 7, pp. 4–4, 2007.
- [27] M. Cova, C. Kruegel, and G. Vigna, “Detection and analysis of drive-by-download attacks and malicious javascript code,” in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 281–290.
- [28] A. K. Sood and R. J. Enbody, “Malvertising—exploiting web advertising,” *Computer Fraud & Security*, vol. 2011, no. 4, pp. 11–16, 2011.
- [29] T. R. Peltier, “Social engineering: Concepts and solutions,” *Information Systems Security*, vol. 15, no. 5, pp. 13–21, 2006.
- [30] C. Smutz and A. Stavrou, “Malicious pdf detection using metadata and structural features,” in *Proceedings of the 28th annual computer security applications conference*. ACM, 2012, pp. 239–248.
- [31] A. Giani, V. H. Berk, and G. V. Cybenko, “Data exfiltration and covert channels,” in *Defense and Security Symposium*. International Society for Optics and Photonics, 2006, pp. 620 103–620 103.
- [32] S. J. Murdoch and S. Lewis, “Embedding covert channels into tcp/ip,” in *Information hiding*, vol. 3727. Springer, 2005, pp. 247–261.
- [33] S. Zander, G. Armitage, and P. Branch, “A survey of covert channels and countermeasures in computer network protocols,” *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 44–57, 2007.
- [34] “Tempest for eliza,” <http://www.erikyyy.de/tempest/>, 2018, (Accessed on 12/03/2017).
- [35] “funtenna - github,” <https://github.com/funtenna>, 2016, (Accessed on 14/06/2018).
- [36] M. Guri, M. Monitz, and Y. Elovici, “Usbee: Air-gap covert-channel via electromagnetic emission from usb,” in *Privacy, Security and Trust (PST), 2016 14th Annual Conference on*. IEEE, 2016, pp. 264–268.
- [37] M. Guri, B. Zadov, and Y. Elovici, “Odini: Escaping sensitive data from faraday-caged, air-gapped computers via magnetic fields,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1190–1203, 2019.
- [38] M. Guri, A. Daidakulov, and Y. Elovici, “Magneto: Covert channel between air-gapped systems and nearby smartphones via cpu-generated magnetic fields,” *arXiv preprint arXiv:1802.02317*, 2018.
- [39] M. Guri, B. Zadov, D. Bykhovsky, and Y. Elovici, “Powerhammer: Exfiltrating data from air-gapped computers through power lines,” *IEEE Transactions on Information Forensics and Security*, 2019.
- [40] ———, “Ctrl-alt-led: Leaking data from air-gapped computers via keyboard leds,” in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1. IEEE, 2019, pp. 801–810.
- [41] M. Guri, B. Zadov, A. Daidakulov, and Y. Elovici, “xled: Covert data exfiltration from air-gapped networks via switch and router leds,” in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2018, pp. 1–12.
- [42] M. Guri and D. Bykhovsky, “air-jumper: Covert air-gap exfiltration/infiltration via security cameras & infrared (ir),” *Computers & Security*, vol. 82, pp. 15–29, 2019.
- [43] M. Guri, O. Hasson, G. Kedma, and Y. Elovici, “An optical covert-channel to leak data through an air-gap,” in *Privacy, Security and Trust (PST), 2016 14th Annual Conference on*. IEEE, 2016, pp. 642–649.
- [44] M. Guri, “Optical air-gap exfiltration attack via invisible images,” *Journal of Information Security and Applications*, vol. 46, pp. 222–230, 2019.
- [45] B. Carrara and C. Adams, “On acoustic covert channels between air-gapped systems,” in *International Symposium on Foundations and Practice of Security*. Springer, 2014, pp. 3–16.
- [46] M. Guri, Y. Solewicz, and Y. Elovici, “Fansmitter: Acoustic data exfiltration from air-gapped computers via fans noise,” *Computers & Security*, p. 101721, 2020.
- [47] M. Guri, Y. Solewicz, A. Daidakulov, and Y. Elovici, “Acoustic data exfiltration from speakerless air-gapped computers via covert hard-drive noise (diskfiltration),” in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 98–115.
- [48] M. Guri, “Air-viber: Exfiltrating data from air-gapped computers via covert surface vibrations,” *arXiv preprint arXiv:2004.06195*, 2020.
- [49] J. Dean, “Jumping the airgap,” <https://thoughtworksnc.com/2017/03/16/jumping-the-airgap/>, 03 2017, (Accessed on 02/27/2018).
- [50] K. H. Billings and T. Morey, *Switchmode power supply handbook*. McGraw-Hill, 2011.

- [51] J. von Kistowski, H. Block, J. Beckett, C. Spradling, K.-D. Lange, and S. Kounev, "Variations in cpu power consumption," in *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*. ACM, 2016, pp. 147–158.
- [52] "pthread_mutex_lock(3): lock/unlock mutex - linux man page," https://linux.die.net/man/3/pthread_mutex_lock, (Accessed on 12/03/2017).
- [53] "Praat: doing phonetics by computer," <http://www.fon.hum.uva.nl/praat/>, 2018, (Accessed on 02/27/2018).
- [54] D. Marr, F. Binns, D. Hill, G. Hinton, D. Koufaty *et al.*, "Hyper-threading technology in the netburst® microarchitecture," *14th Hot Chips*, 2002.
- [55] <https://cryptome.org>, "Nstissam tempest/2-95," <https://cryptome.org/tempest-2-95.htm>, 2000, (Accessed on 02/27/2018).
- [56] B. Carrara and C. Adams, "A survey and taxonomy aimed at the detection and measurement of covert channels," in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2016, pp. 115–126.
- [57] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proceedings of the 6th ACM symposium on information, computer and communications security*. ACM, 2011, pp. 355–366.
- [58] "Products - pulsar instruments plc," <https://pulsarinstruments.com/en/categories>, 2018, (Accessed on 06/14/2018).
- [59] "Audio jammer — counter surveillance systems," <https://www.brickhousesecurity.com/counter-surveillance/audio-jammers/>, 2018, (Accessed on 06/14/2018).