Covert Electromagnetic Transmission using Raspberry Pi CPU

Technical report

Year	2025
Institution	Fontys University of Applied Sciences
Faculty	Information & Communications Technology (ICT)
Tracks	Open Learning, Delta
Stakeholder	Cybersecurity Research Group
Students	Artur Kraskov, Hitesh Manglani, Antara Frisch, Nahit Sogulu, Patryk Olejnik, Kelly Zhang
Mentors	Mark Madsen, Casper Schellekens, Ousta Firas, Edwin van den Oetelaar, Jan Dobbelsteen

Contents

Contents **1**
Introduction **1**
Literature Review **2**
Covert channels 2
EMC testing of Raspberry Pi 2
Methodology **2**
Preliminary phase & MVP 2
Equipment list 3
Initial Python scripts 4
Ranges & modulation 4
C based prototype 5
Software-Based Detection System 7
CPU monitoring(Nahit) 7
USB monitoring (Kelly) 8
Steganography 8
Story and Accessibility 9
Phase 2: Night of the Nerds Presentation 10
Results **11**
Results with MVP 11
Results with C based prototype (upgraded version) 12
Results with software-based detection system 13
Discussion **13**
What is currently visible as a way forward for the offensive side? How to improve the attack? 13
What are the recommendations for the defensive side? 14
Software based defense system (Antivirus) 14
CPU monitoring 14
USB monitoring 14
Conclusion **14**
Reference **15**

Introduction

While air-gap isolation has traditionally been considered a robust security measure, recent studies demonstrate that even air-gapped devices emit exploitable EM signals. Research extensively documents covert channels based on CPU load, memory bus activities, and power supply emissions. However, these studies largely overlook the extensive deployment of low-power, compact devices like the Raspberry Pi, despite their proliferation in sensitive IoT and embedded system environments.

Given the unique hardware features of the Raspberry Pi—including low-power ARM CPUs, DDR memory architecture, and extensive GPIO interfaces—this platform represents an important frontier for covert channel research.

Literature Review

Covert channels

Current research extended on preliminary study by one of the students [12]. And involved extensive research of existing covert channel attacks, external electromagnetic monitoring methods, electromagnetic compliance testing and standards, signals.

Guri M. "Air-Gap Electromagnetic Covert Channel" [2] paper provided foundational insights into general covert-channel attack and into using power supply fluctuations with simple core-based busy-wait loops for OOK modulation and calibration schemes.

Guri M. *"RAMBO: Leaking Secrets from Air-Gap Computers by Spelling Covert Radio Signals from Computer RAM"* [1] formed our knowledge basis for further exploration of cache-eviction tests on the Raspberry Pi 4.

The BitJabber study [3] led to an attempt to implement Frequency Shift Keying (FSK) using two distinct frequencies, following a similar attack strategy to that of the RAMBO attack. However, the frequency shifts were not stable enough to be used reliably, so this method was not included in the final design.

EMC testing of Raspberry Pi

Notably there were not many publications found about existing Raspberry Pi 4 EMC facility testing. One of the sources provided information about particular spectrum ranges with stronger signals and hardware architecture components that produced EMF in these ranges [13]. This information allowed us to speed up prototyping.

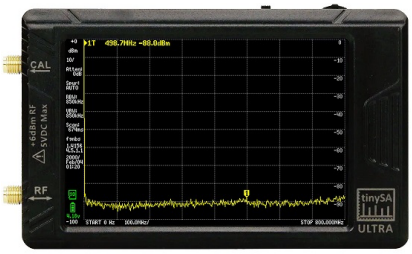
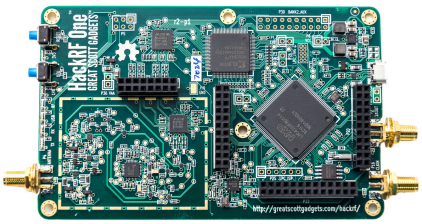
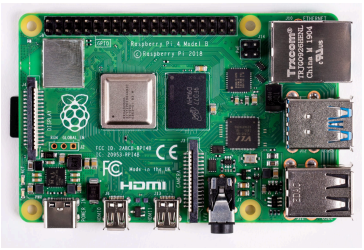
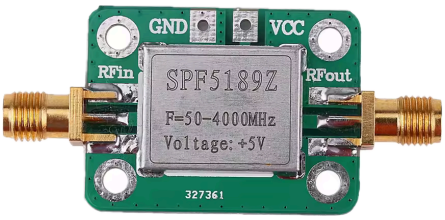
Methodology

Preliminary phase & MVP

How can we know that device is not sending or transmitting anything? How can we know that device is even there? What if it is air gapped or not using any networks? IoT devices populate modern environments. Existing practices in bug detection, signal intelligence and spectrum analysis shape the framework. Preliminary study lists all steps leading from a problem to a lab project with a student team [12]. Began with simple spectrum analyzer, low-noise amplifier (LNA) and near-field probes to detect EMF from Raspberry Pi 4B.

Equipment list

All images and links are taken from internet public resources that appeared in DuckDuckGo.

Photo	Description	Link
	SIGLENT SSA 3032X Spectrum Analyzer	https://siglentna.com/product/ssa3032x/
	TinySA Portable Spectrum Analyzer	https://www.tinysa.org/wiki/
	HackRF One Software Defined Radio (SDR)	https://greatscottgadgets.com/hackrf/one/
	Raspberry Pi 4 Model B	https://www.raspberrypi.com/products/raspberry-pi-4-model-b/
	10kHz-6GHz LNA amplifier was used (similar can be used, primitive from aliexpress used)	https://reversepcb.com/the-basics-of-low-noise-amplifiers/
	Near-field probes (used primitive from aliexpress)	https://www.allaboutcircuits.com/technical-articles/introduction-to-near-field-probes-and-their-use-in-emc-troubleshooting/

	Coaxial cables (used primitive from aliexpress)	https://www.conwire.com/blog/coaxial-cable-guide/

Equipment list

Initial Python scripts

Python script utilizing SDR instead of spectrum analyzer allows to significantly reduce costs (around 10 times from SA - 3000$ to SDR - 200-300$). As well, switching from Spectrum Analyzer to SDR allowed programmatic access to data. SIGLENT Spectrum Analyzer native software is more complicated and less accessible than hackrf available as Python library and supported by GQRX, GNU Radio and other open source software tools.

A collection of scripts on Github (load CPU/RAM, read and visualize sweep data, use GPIO for RF transmission, etc.):
https://github.com/Krasnomakov/cs_research/tree/research/Research

Ranges & modulation

Focused on 200-300MHz due to stronger signals in that range possible to capture with available equipment and antennas. Most cheap Log Periodic antennas work 100MHz up. Also several noisy ranges were specified in Vaclav's EMC research [13].

The expected problematic frequencies are following:

- 24MHz (clock out provided to connect additional USB Hubs),
- 25 MHz (crystal for Ethernet connectivity),
- 60 MHz (camera),
- 250 MHz (GPU-Graphic Processing Unit),
- 340 MHz (HDMI-High-Definition Multimedia Interface),
- 450 MHz (SDRAM-Synchronous Dynamic Random Access Memory),
- 900 MHz (ARM-Advanced RISC Machines).

Problematic frequency ranges [13]

C based prototype

To enhance the preliminary Python-based implementation, a C-based prototype was developed to address key problems. The original system relied on GQRX, which required manual setup, introduced delays, and handled all signal processing internally, making it difficult to customize or troubleshoot.

Python's timing controls were also too imprecise for reliable synchronization, and performance was limited by GQRX's overhead. By switching to C, we gained precise timing with clock_nanosleep, better performance using FFTW and threads, and full control over signal acquisition, processing, and visualization.

The system was split into two main components: a transmitter that generated electromagnetic emissions, and a receiver that captured and decoded those signals.

Transmitter
The transmission hardware was a Raspberry Pi 4 (our target device), which uses a built-in Power Management IC (PMIC) to dynamically adjust power delivery through DVFS (Dynamic Voltage and Frequency Scaling). By placing the CPU under load, we could manipulate the PMIC's behavior to emit detectable electromagnetic signals. The transmitter toggled all four CPU cores using tight busy-wait loops to modulate data bits using On-Off Keying (OOK).

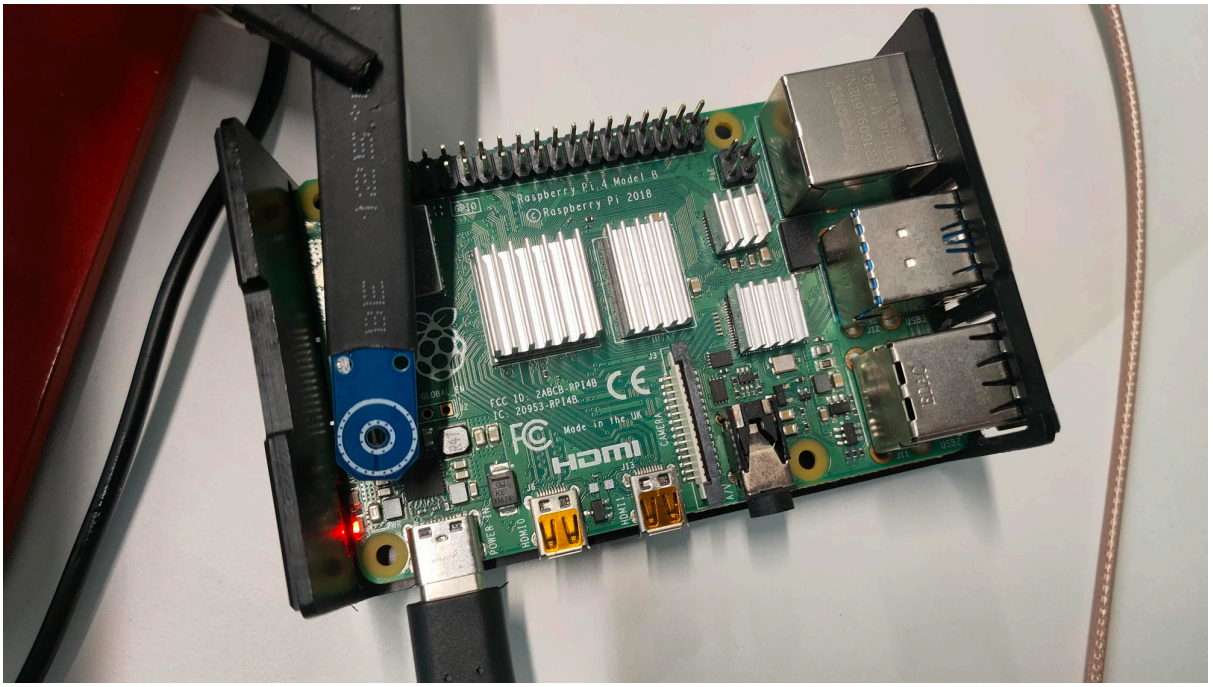


Fig 1. A Near Field Probe on top of the Transmitter's (Raspberry PI 4's) Power Management IC

To evaluate the feasibility of RAMBO-style cache eviction attacks, we used Linux's perf tool to monitor hardware-level events while inducing high cache miss rates through specific memory access patterns

Perf confirmed the method's effectiveness, raising cache miss traffic from roughly 2.9 million misses per second to 6.0 million misses per second, a 105 percent increase (Table 1). However, the resulting spectrum was neither stable nor distinct enough to decode, and appeared more like background noise from the power supply.

After consulting a hardware specialist, it was suggested that the Pi's multilayer PCB and compact internal wiring may suppress electromagnetic leakage from the memory bus. That said, measurement error or issues in our setup cannot be ruled out.

Loop Stride	**Time (s)**	**Total Cache Refs**	**Total Cache Misses**	**Misses/s**
Stride = 1	10.30	3,148,427,713	30,443,413	2.96 M
	15.20	4,696,627,568	44,900,314	2.95 M
	20.10	6,231,480,002	58,133,942	2.89 M
Avg (Stride 1)	-	-	-	**2.93 M**
Stride = 64	10.08	3,067,688,968	56,828,097	5.64 M
	15.30	4,601,396,916	88,941,007	5.81 M
	20.20	6,099,336,825	133,750,881	6.63 M
Avg (Stride 64)	-	-	-	**6.02 M**

Table 1. Comparing loop strides of 1 and 64 bytes. A 64 byte loop stride would effectively cause cache misses.

Receiver

The receiver used a near-field probe connected to a low-noise amplifier (LNA) to boost the weak EM signals from the Pi. The LNA had a 0.4 dB noise figure and 20 dB gain at 1.95 GHz and was used with its default settings.

FFT processing utilized FFTW with a size of 262,144 samples concatenated twice per reading for enhanced noise reduction. Bit detection relied on identifying significant magnitude shifts with ±15% thresholds to decode binary signals. Timing synchronization was also taken into consideration: the transmitter initiated at exact whole-second boundaries, and the receiver began precisely 100 ms later, both tolerating up to 10 ms jitter.

To improve synchronization and achieve reliable detection of the actual data stream, a simple calibration system was added, inspired by the "Air-Gap Electromagnetic Covert Channel" [2] paper. The receiver continuously scans and decodes incoming bits, and

begins only interpreting them as valid data once it detects a predefined synchronization pattern “10101010”. Once this pattern is matched, the receiver transitions into “real” data capture and packet re-constructing until an explicit “END of symbol” is received. This ensures the receiver knows exactly when to start and stop decoding.

Software-Based Detection System

For making the Software-based defense system we began by analyzing which types of malware have been found in air-gapped systems. We divided our findings into two categories [8]:

1) **Previously Detected Malware in Air-Gapped Systems**
   These are real-world examples of malware that have been discovered and documented in air-gapped environments, such as Ramsay, USBCulprit, and others.

2) **Experimental Malware Examples**
   These are conceptual or prototype malware described in academic research papers, based on electromagnetic, magnetic, acoustic, or optical transmission techniques.

For each malware example, we describe its characteristics and explain what it does within the air-gapped environment.

We also analyzed different types of defense systems. For each defensive element, we discuss the advantages and disadvantages in the context of protecting air-gapped systems.

Based on this analysis, we concluded that using a combination of hardware- and software-based detection systems provides the strongest resilience.

Lastly, we provided advice for how the software-based defense system should be designed, based on a set of detection rules [9].

CPU monitoring

A big part of electromagnetic emissions have a visible impact on the CPU[9]. In the current transmitter, the generated EM signals can be monitored by looking at the CPU consumption of the transmitter. The project in its current form can read these consumption metrics and store them in a log file[8]. For the future, machine learning models that perform anomaly detection could help a lot in the detection system. An anomaly detection approach was chosen for its robustness in detecting different types of novel EM exfiltration techniques that has different impacts on the CPU but it has a downside which is that the normal CPU consumption metrics should be sampled from a realistic environment. Ideally, the end user should be able to train the machine learning model in their live environment to reduce false positives and overfitting.

USB monitoring

Based on the recommendation from the software-based detection system:

The system should detect data being written to a USB device from the air-gapped machine, because malware present on such a system can replicate itself and exfiltrate data using USB flash drives to steal sensitive information such as credentials, screenshots, and more. This type of attack is often the only viable method of both infection and data exfiltration in air-gapped environments.[9]

As a result, a USB monitoring system was developed that continuously scans for newly connected USB devices using the lsblk command. This is done by generating a unique key for each connected USB device, which consists of: the device name, the number of partitions and a hash of all mount points on the USB stick.

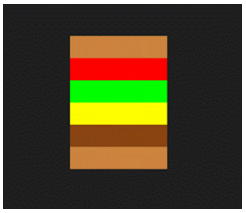
By comparing these keys over time, the system can detect: Which USB sticks have been newly plugged in and Any changes in the number of mount points or partitions — a technique that can be exploited by malware to exfiltrate data[10][11].

When a USB device is detected, the following actions on files are monitored on the device on each mountpoint using fanotify:

- Open/read
- Write
- Create
- Delete
- Moved from / moved to
- Rename
- Attribute change
- Write close / read close

Steganography

As a potential extension of covert electromagnetic transmission, this project explored the idea of using steganography to embed short messages into small image files. A custom Python script was used to generate a 4×6 pixel image resembling a burger and embed a message ("404 sauce") by modifying the least significant bits (LSBs) of the RGB values. A corresponding decoder script was written to extract the message on the receiver's end.



Steganography 4 x 6 pixel burger

Due to the current system's very low data transfer rate (~2 bits per second), this method could not be implemented in practice. However, it remains a viable future direction once faster hardware is available

This code can be found in the following git, in the Branch Steganography [8].

Story and Accessibility

Introduction

Raising awareness about cybersecurity threats is a key mission of the Fontys Cybersecurity Research Group. This project aimed to contribute to that mission by combining technical research with creative communication. The objective was not only to investigate the feasibility of electromagnetic (EM) data exfiltration from air-gapped systems but also to present the findings in a way that would engage and inform a broader audience. By making the topic accessible through storytelling, the project fulfilled both an educational and awareness-raising function.

Phase 1: Story Development

From the outset, the project team recognized the importance of accessibility in communicating complex cybersecurity topics. Following initial feedback emphasizing the need for a more engaging presentation, and in preparation for participation in the Night of the Nerds event, the team explored ways to improve the storytelling aspect of the project.

After considering several ideas, the team was inspired by the SpongeBob SquarePants "Krabby Patty" storyline. This familiar and humorous theme provided an ideal foundation for a narrative-driven concept that would resonate with a wide audience. A supporting hardware design and a poster were created to reinforce the theme and enhance the visual appeal of the project.

This poster follows the following storyline:

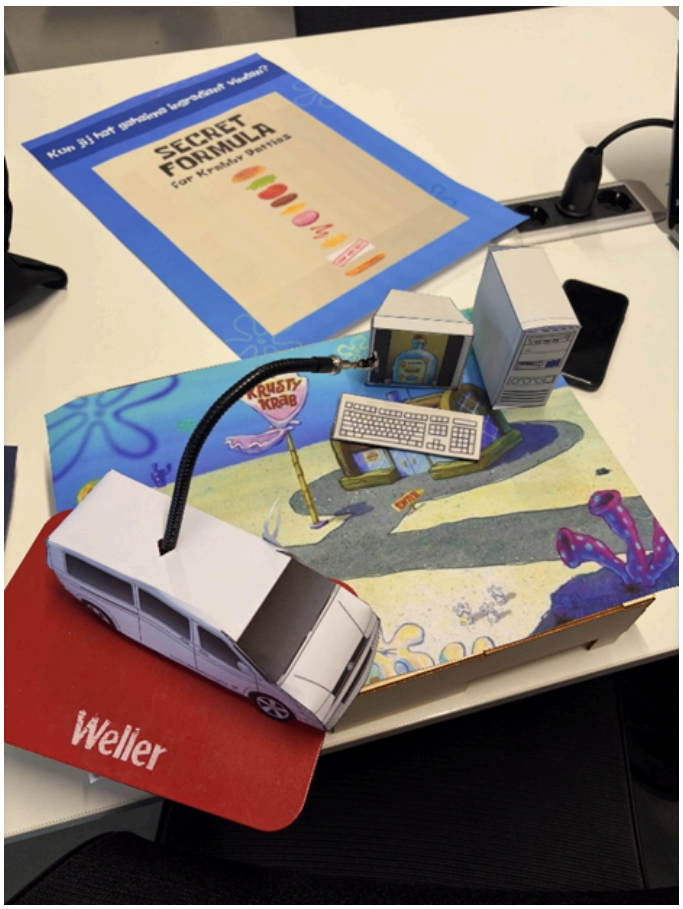
Plankton wants to steal the Krabby Patty secret recipe, which is stored on a computer that isn't connected to the internet (air-gapped). Knowing this makes digital theft more difficult, he recruits Squidward, who is tired of his job, to plant a virus in the system. As the virus begins downloading the secret recipe using covert signals, SpongeBob notices in time and disrupts the transmission. In the end, Plankton and Squidward are caught, and the secret recipe remains safe.

Air-gapped story poster

This creative approach was well received by both technical and non-technical audiences. It allowed the team to explain a highly technical topic, covert EM data transmission using a Raspberry Pi 4, in a way that was understandable, entertaining, and educational. By combining rigorous research with accessible storytelling, the project succeeded in raising cybersecurity awareness in a format aligned with the Fontys Cybersecurity Research Group's mission.

Phase 2: Night of the Nerds Presentation

To enhance interactivity for the Night of the Nerds event, the "Burger Treasure Hunt" was introduced. Participants used a spectrum analyzer to detect EM signals above the Raspberry Pi's power supply, revealing encoded burger ingredients, ultimately uncovering the secret Krabby Patty recipe. This interactive experience encouraged curiosity and hands-on exploration, making it a standout feature of the exhibition. The theme was visually reinforced using paper models inspired by the SpongeBob universe, as shown in the accompanying image.



Initial design for Night of the Nerds.

The setup features a flipped box decorated with SpongeBob-themed wallpaper, concealing all hardware and cables. A paper model PC houses the transmitting Raspberry Pi, while a paper spy van, complete with a tiny Plankton figure, holds the spectrum analyzer, enhancing the theme and interactivity.

Results

Results with MVP

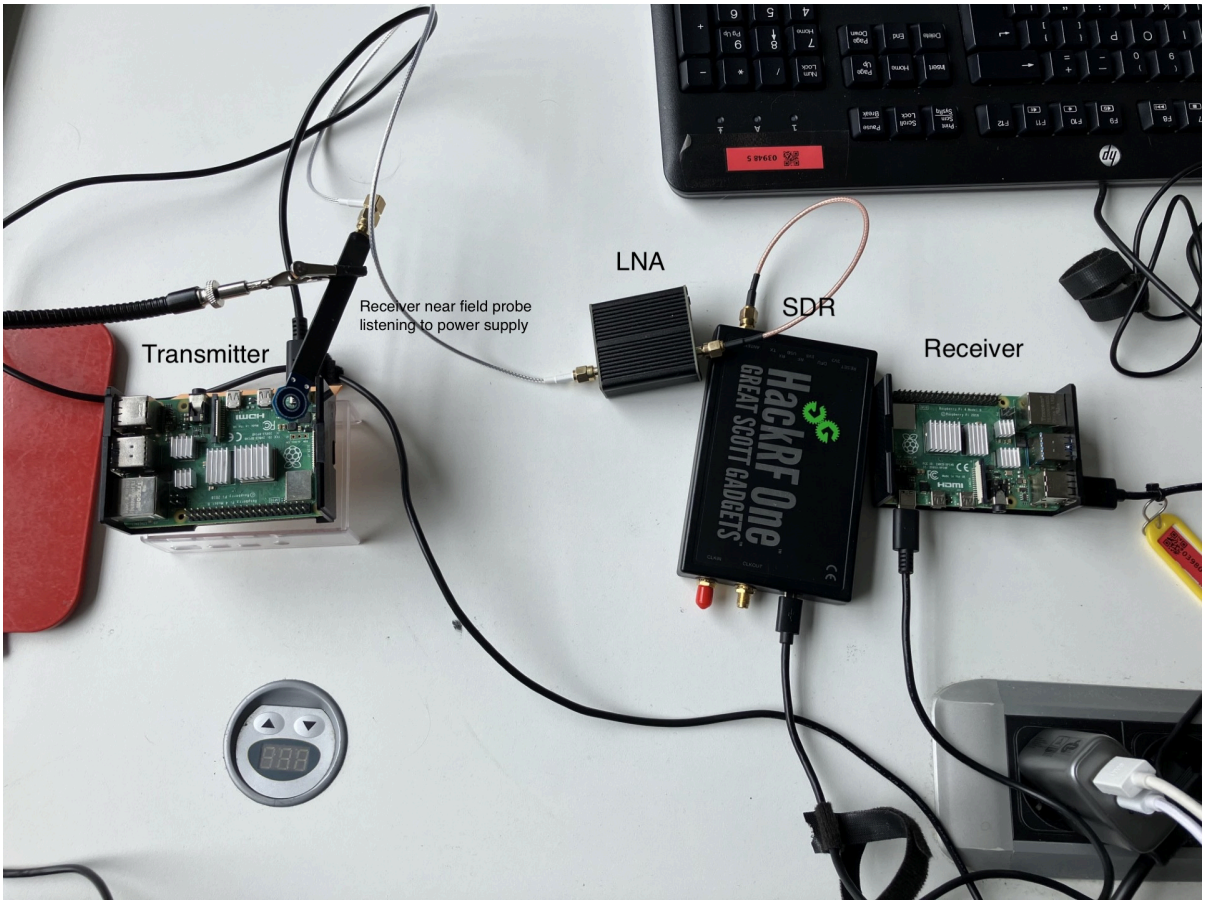
Realized working pair (transmitter/receiver) capable of encoding, transmitting and decoding binaries “0101010” . Used Raspberry Pi CPU as a controlled element and power supply as main source of EMF. Frequency ranges around 300MhZ. Uses C script to produce modulation.

Receiver employed HackRF One SDR, LNA, near-field probe, used raspberry pi, but can run on PCs or intel/arm based laptops as well. Uses GQRX and Python for decoding.

Realized prototype allowed easy entry point and demonstrable process - air gap communication.

Ewo simulations (transmitter/receiver) using log and .wav file for encoding and decoding and different types of modulation (OOK, MFSK) were .

Demo: https://vimeo.com/1075115790
Github: https://github.com/Krasnomakov/cs_research/tree/research/MVP



Working pair MVP set up (photo from Github)

Results with C based prototype (upgraded version)

The final C-based implementation was able to transmit data at a rate of 2 bits per second with an estimated accuracy of approximately 90 percent. Reliable signal detection was achieved at a distance of around 10 centimeters using a near-field probe and up to 1 meter with a loop antenna. Although formal Bit Error Rate analysis is still pending, testing under controlled conditions showed the system worked consistently.

The signal was visualized using GQRX during development, and manual testing confirmed that the transmitted bits matched the received output. A graphical user interface was also developed to present a live decoding demo during the school presentations.

Challenges

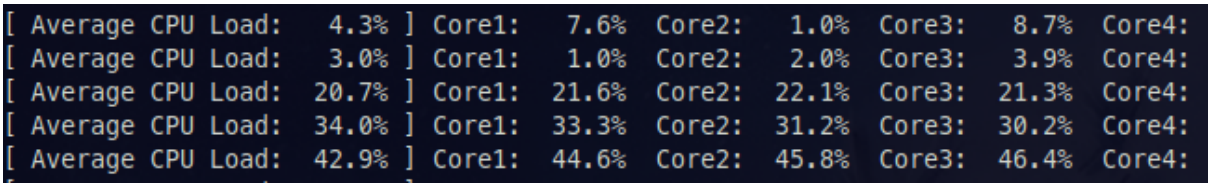
One of the main technical challenges was achieving accurate time synchronization between the transmitter and receiver. This was addressed by having the transmitter send data exactly at the start of each second, while the receiver began sampling 100 milliseconds later to account for timing offsets. Another challenge was the low intensity of electromagnetic emissions from the Raspberry Pi.

Resolutions

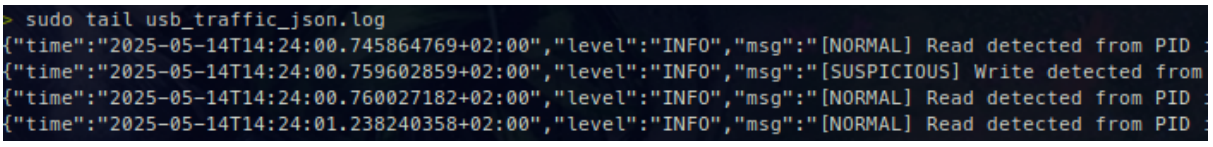
To overcome the weak signal strength and improve decoding reliability, the receiver was enhanced in several ways. FFT size was increased to improve frequency resolution and multiple samples per symbol were averaged to reduce noise. The system retained a simple OOK (On-Off Keying) modulation scheme for clean signal separation. Additionally, the FFT backend was switched from KISSFFT to FFTW, which increased the speed of signal processing.

Results with software-based detection system

The software-based detection system has made an initial step in detecting two types of attacks. The first involves high CPU load(above 90%), which is commonly used in electromagnetic transmissions, as described in the category *Experimental Malware Examples*.



Additionally, the system is capable of detecting data exfiltration via USB flash drives, which is the most frequently used method by malware listed under *Previously Detected Malware in Air-Gapped Systems*, as identified in the analysis of the software-based detection system.



Discussion

What is currently visible as a way forward for the offensive side? How to improve the attack?

Accurate time alignment between the transmitter and receiver was essential for reliable decoding. The transmitter was set to send data at the start of each second, while the receiver began sampling 100 milliseconds later to account for any timing mismatch. Since the Raspberry Pi produced very weak electromagnetic signals, the

receiver relied on large FFT sizes, averaging, and carefully tuned thresholds to detect bit changes. Even with these adjustments, performance was limited by the low signal strength, suggesting that better receiver hardware and a stronger transmission source would significantly improve results.

While the system reached a transmission rate of 2 bits per second with about 90 percent accuracy, longer tests showed that frequency drift became an issue as the Pi heated up. Future improvements could include adaptive frequency tracking, active cooling, and testing with more capable hardware. Increasing the EM output from the transmitter and using a more sensitive receiver could enable much faster and more reliable data transmission.

What are the recommendations for the defensive side?

Software based defense system (Antivirus)

CPU monitoring

An anomaly detection approach was chosen for its robustness in detecting different types of novel EM exfiltration techniques that has different impacts on the CPU but it has a downside which is that the normal CPU consumption metrics should be sampled from a realistic environment. Ideally, the end user should be able to train the machine learning model in their live environment to reduce false positives and overfitting.

USB monitoring

The USB monitoring system currently only includes the detection of newly connected USB sticks and the monitoring of various file-level actions.

As a next step, a prevention mechanism should be implemented to block certain actions, such as write operations, and to reduce false positives.

This can be achieved in several ways, for example:

- By blocking specific actions using system commands
- Or by using fanotify with a different flag that allows interception and prevention of events

Conclusion

This work establishes the ground layer with practical implementations for group or personal research of electromagnetic covert channels. It bridges the gap between science and industry by providing ICT students with necessary information and all relevant tools and materials for further research and development projects. It is nicely packed into popular culture stories, published on github, has readme, photos and video. List of equipment is very basic and relatively cheap, and can be obtained worldwide with aliexpress for less than

100$. Technical report includes all relevant references and comments and backs up the findings with scientific data. The research bridges the gap in the area of air gapped microcomputers and shows how the electromagnetic covert channel can be realized with raspberry pi and software defined radio.

Reference

1. Guri M. RAMBO: Leaking Secrets from Air-Gap Computers by Spelling Covert Radio Signals from Computer RAM.
   https://arxiv.org/pdf/2409.02292
2. Guri M. Air-Gap Electromagnetic Covert Channel. IEEE.
   https://ieeexplore.ieee.org/document/10197447
3. BitJabber: The World's Fastest Electromagnetic Covert Channel
   https://zhenkai-zhang.github.io/papers/bitjabber.pdf
4. Python SDR Tutorial.
   https://pysdr.org/
5. GNU Radio Foundation. What is Software Defined Radio?
   https://wiki.gnuradio.org/index.php/What_is_GNU_Radio%3F
6. What is Modulation? Why Modulation is Required? Types of Modulation Explained.
   What is Modulation ? Why Modulation is Required ? Types of Modulation Expla…
7. The Air-Gap Jumpers
   The Air-Gap Jumpers
8. Github page
   https://github.com/HiteshManglani123/air-gapped.git
9. Software based defense system document
   air-gapped/documentation/detection system airgap.pdf at dev · HiteshManglani123/air-gapped
10. USB monitoring realisation document
    air-gapped/documentation/usbmonitoring_realisatie.pdf at dev · HiteshManglani123/air-gapped
11. USB monitoring design document
    air-gapped/documentation/USBmonitoring-desgin.pdf at dev · HiteshManglani123/air-gapped
12. Kraskov A. (2025). Technical report: Digital Forensics in IoT security
    https://github.com/Krasnomakov/cs_research/blob/main/publications/covert_channels_technical_report.pdf
13. Vaclav et al. (2017). Electromagnetic compatibility of Raspberry Pi development platform in near and far-field
    https://ieeexplore.ieee.org/document/8293550