

# Wanna TEMPEST your computer?

Florian Barbarin, Maxime Gagliardini and Guillaume Squillaci

TLS-SEC

March 13, 2018



**TLS-SEC**

**Toulouse**  
**Hacking**  
**Convention**

# Table of contents

Introduction

Transmitter

Receiver

Data exfiltration

Conclusion

# "Projet long" context

## THC 2018 Challenge

- Prepare tutorial challenge
- Work based on *GSMem: Data Exfiltration from Air-Gapped Computers over GSM Frequencies* (24th USENIX Security Symposium)
- "Challenge" : data exfiltration from an air-gapped computer
- "Tutorial" : guide the challenger step-by-step
- Main idea : follow how we succeed to reproduce a part of the paper

# Air-gapped networks



# Air-gapped networks



# Electromagnetic emanations

## Emanations

- Each electronic device has emanations
- Could be optical, electronical, acoustical, mecanical or electromagnetical
- Focus on electromagnetic emanations

## TEMPEST

- Name given by NSA for standards protecting against electromagnetic emanations
- Context : EMSEC, surbpart of COMSEC

# Challenge context

## Goal

Get a password stored on the air-gapped computer

## Problem

Air-gapped computer  $\Rightarrow$  no possibility to gain access and/or exfiltrate data via network

## Solution

Use electromagnetic emanations to create a covert channel and exfiltrate data

# Technical environment

## Devices

- 1 air-gapped computer (attacked computer)
- 1 standard computer (attacker's computer)

## Tools

- Spectrum analyzer
- Software-defined radio : USRP/RTL-SDR
- Antennas
- Softwares : URH/GNURadio



# Table of contents

Introduction

Transmitter

Receiver

Data exfiltration

Conclusion

# Transmission without specific component

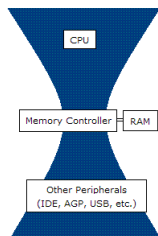
## Our target (remind)

Use electromagnetic emanations to create a covert channel and exfiltrate data

## Problems

- Computer's electromagnetic emanations  $\rightarrow$  low amplitude
- Amplitude increase  $\Rightarrow$  circuit tension increase

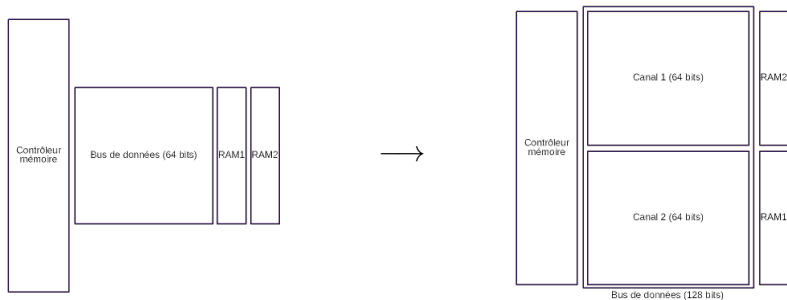
# Problem bypass : Multi-channel memory architectures



## Memory access

- Memory controller → bottleneck
- Intel/AMD → Multi-channel memory architectures
- Increase data bus size :
  - Double channel  $\Rightarrow$  128 bits
  - Triple channel  $\Rightarrow$  192 bits
  - Quadruple channel  $\Rightarrow$  256 bits

## Problem bypass : Multi-channel memory architectures

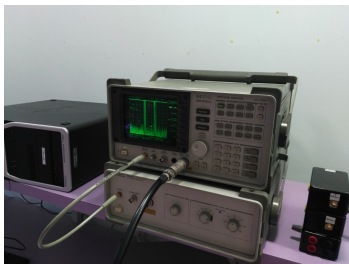


### Consequence using Multi-channel memory

- More electrons in movement at the same time
- Significant electromagnetic emanations created
- $\Rightarrow$  Emanations could be used to create covert channel

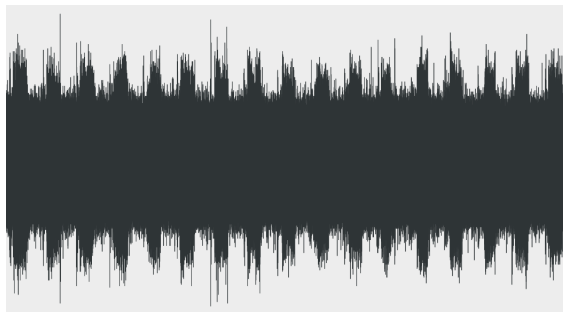
# Problem bypass : Multi-channel memory architectures

## Application



- Be sure to bypass CPU optimizations
- Use of MOVNTDQ instruction on `xmm` registers
- Emanations highlight :
  - Find emission frequency : spectrum analyzer
  - Watch signal : USRP/RTL-SDR + URH

# Problem bypass : Multi-channel memory architectures



## Modulation

- Modulation = information coding process
- Binary Amplitude-Shift Keying (B-ASK) modulation :
  - bit 0  $\leftrightarrow$  normal emission level
  - bit 1  $\leftrightarrow$  average emission level when multi-channel memory is used

# Table of contents

Introduction

Transmitter

Receiver

Data exfiltration

Conclusion

# Demodulation



## GNUradio

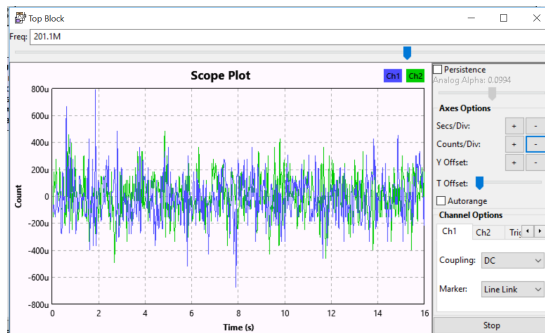
- Free and Open-source Software
- Software Radio
- Signal Processing

## Configuration

- Sample Rate
- Frequency
- Period
- Gain



# Raw Signal

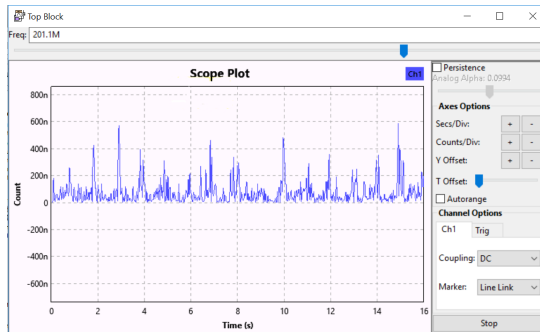


## Source Output

- Complex Signal:
  - In-phase
  - Quadrature

$$s(t) = i(t) + jq(t) = r(t) \cdot e^{j\varphi(t)}$$

# Instantaneous Power

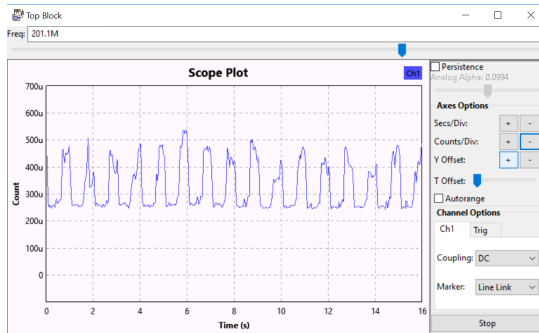


## Float Signal

- Amplitude
- Magnitude

$$P(t) = |r(t)|^2$$

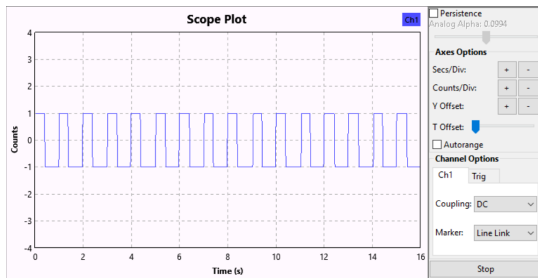
# Sliding Average



## Float Signal

- Number of samples
- Threshold

## Signal Before Decision



### Square Wave

- Clock period
- Output Byte
- Byte Processing

# Signal processing

## Things to improve

- Noise
- Synchronization
- Transmitter
- BER

# Table of contents

Introduction

Transmitter

Receiver

Data exfiltration

Conclusion

# The Malware

## Idea

- Same skeleton than the Transmission section
- 1bit/s to reduce error rate

## What is transmitted ?

- The message : a file in argument
- End of File : 00001010
- A pattern to localize the message : 11111111
- All of this in an infinite loop

# Reception

## In GNU Radio

- Start the malware before the acquisition to avoid a bug in the first bit transmitted
- Wait around 2 minutes

## Conversion of the GRC file into our flag

- Localize a first pattern
- Extract bytes from this pattern until the next one
- Convert extracted bytes into ASCII and print it



# Reception

```
maxime@Max-PC:~/Github/THC-Projet-long---TEMPEST-attacks/Receiver$ xxd -b pack_bytes | grep "1111111 1"
00000000: 00000101 01111111 10111010 00011001 10110110 10111000 .....
00000006: 00011001 10111001 10111010 00000101 01111111 10111010 .....
```

```
maxime@Max-PC:~/Github/THC-Projet-long---TEMPEST-attacks/Receiver$ ./a.out pack_bytes
bin = 0000010101111111011101000011001101101101011100000011001101110011011101000000101011111110111010
msg = 0111010000110011011011010111000000110011011100110111010000001010
ascii = t3mp3st
```

# Reception

```
void extract_msg(FILE * fic, char * msg, char * pattern)
{
    char chaine[TAILLE_MAX],
        tmp1[TAILLE_MAX],
        *tmp2,
        tmp3[TAILLE_MAX]="",
        *tmp4,
        bin[TAILLE_MAX]="";

    while (fgets(chaine, TAILLE_MAX, fic) != NULL)
    {
        memset (tmp1, 0, sizeof (tmp1));
        strncpy(tmp1, chaine+10, 55);
        rem_space(tmp1);
        strcat(bin, tmp1);
    }
    tmp2 = strstr(bin, pattern);
    strcpy(tmp3, tmp2+strlen(pattern));
    tmp4 = strstr(tmp3, pattern);
    strncpy(msg, tmp3, tmp4-tmp3);
}
```

# Table of contents

Introduction

Transmitter

Receiver

Data exfiltration

Conclusion

# Conclusion

## Researchers work

- Better equipment
- Signal processing optimization
- Reception until 30m

## Limits

- Very limited flow rate to prevent error (error detection ?)
- Can be improve with a frequency modulation (attack USBee)

## Countermeasures

- Faraday cage
- Zonal approach
- Antivirus